

AFIPS

CONFERENCE PROCEEDINGS

VOLUME 39

1971

FALL JOINT COMPUTER CONFERENCE

AFIPS PRESS
210 SUMMIT AVENUE
MONTVALE, NEW JERSEY 07645

November 16-18, 1971
Las Vegas, Nevada

The ideas and opinions expressed herein are solely those of the authors and are not necessarily representative of or endorsed by the 1971 Fall Joint Computer Conference Committee or the American Federation of Information Processing Societies, Inc.

Library of Congress Catalog Card Number 55-44701

AFIPS PRESS
210 Summit Avenue
Montvale, New Jersey 07645

©1971 by the American Federation of Information Processing Societies, Inc., Montvale, New Jersey 07645. All rights reserved. This book, or parts thereof, may not be reproduced in any form without permission of the publisher.

Printed in the United States of America

CONTENTS

DATA COMMUNICATIONS

A universal cyclic division circuit.....	1	A. W. Maholick R. B. Freeman
Cyclic redundancy checking by program.....	9	P. E. Boudreau R. F. Steen

APPLICATIONS OF COMPUTERS IN EMERGING NATIONS

Development of computer applications in emerging nations.....	17	A. B. Kamman
Notions about installing and maintaining a population register in Brazil...	27	A. L. Mesquita

OPERATING SYSTEMS—SOME MODELS AND MEASURES

The neutron monitor system.....	31	R. Aschenbrenner L. Amiot N. K. Natarajan
A simple thrupt and response model of EXEC 8 under swapping saturation.....	39	J. C. Strauss
Throughput measurement using a synthetic job stream.....	51	D. C. Wood E. H. Forman
A feedback queueing model for an interactive computer system.....	57	G. Nakamura

TERMINALS

Alcoa picturephone remote information system (APRIS).....	65	M. L. Coleman K. W. Hinkelman W. J. Kolechta
Computer support for an experimental PICTUREPHONE® computer system at Bell Telephone Laboratories Incorporated.....	71	E. J. Rodriguez
Proposed braille computer terminal offers expanded world to the blind....	79	N. C. Loeber

SIMULATION OF ENVIRONMENTAL DYNAMICS

Numerical simulation of subsurface environment.....	89	B. L. Bateman D. D. Drew P. B. Crawford
Digital simulation of the general atmospheric circulation using a very dense grid.....	97	W. E. Langlois
Simulation of the dynamics of air and water pollution.....	105	L. W. Ross
Programming the war against water pollution.....	115	D. J. Olsen
Application of a large scale nonlinear programming problem to pollution control.....	123	G. Graves D. Pingry A. Whinston

IMAGES AND PATTERNS

Parametric font and image definition and generation.....	135	A. J. Frank
A syntax-directed approach to pattern recognition and description.....	145	L. D. Menninga
Computer pattern recognition of printed music.....	153	D. S. Prerau

LARGE SCALE INTEGRATION (LSI)

A storage cell reduction technique for ROS design.....	163	C. K. Tang
A new approach to implementing high-density shift registers.....	171	T. S. Jen
Universal logic modules implemented using LSI memory techniques.....	177	K. J. Thurber R. O. Berg

COMPUTERS IN MEDICINE—PROBLEMS AND PERSPECTIVES (PANEL SESSION)

Position paper.....	195	B. G. Lamson
Position paper.....	195	C. T. Post, Jr.
Position paper.....	196	E. E. Van Brundt

THE USER INTERFACE FOR INTERACTIVE SEARCH (PANEL SESSION)

Chairman's Note.....	197	J. L. Bennett
----------------------	-----	---------------

STATE OF THE COMPUTER ART IN BIOLOGY (PANEL SESSION)

Position paper.....	199	C. Levinthal
Position paper.....	199	N. E. Morton
Position paper.....	200	R. Nathan
Position paper.....	201	W. F. Raub
Position paper.....	201	W. S. Yamamoto

SIMULATION OF AEROSPACE SYSTEMS

Introduction to training simulator programming.....	203	D. G. O'Connor
The handling qualities simulation program for the augmentor wing jet STOL research aircraft.....	213	W. B. Cleveland
Software validation of the Titan IIIC digital flight control system utilizing a hybrid computer.....	225	R. S. Jackson S. A. Bravdica
Multivariable function generation for simulations.....	233	P. Chew J. E. Sanford E. Z. Asman

PROGRAMMING LANGUAGES AND LANGUAGE PROCESSORS

Problems in, and a pragmatic approach to, programming language measure- ment.....	243	J. E. Sammet
The ECL programming system.....	253	B. Wegbreit
The "single-assignment" approach to parallel processing.....	263	D. D. Chamberlin
MEANINGEX—A computer-based semantic parse approach to the analysis of meaning.....	271	D. J. Mishelevich

APPLICATION OF COMPUTERS TO LAW ENFORCEMENT AND
CRIMINAL JUSTICE

Law enforcement communications and inquiry systems.....	281	J. D. Hodges, Jr.
The Long Beach public safety information subsystem.....	295	G. Medak P. Whisenand G. Gack
State criminal justice information system.....	303	R. Gallati
Automated court system.....	309	R. Baca M. Chambers W. Pringle S. Boehm

EXPERIMENTS IN ON-LINE DELPHI RESEARCH

Delphi and its potential impact on information systems.....	317	M. Turoff
Technology for group dialogue and social choice.....	327	T. B. Sheridan
Structuring information for a computer-based communications medium....	337	S. Umpleby

INTERACTIVE CONTINUOUS-SYSTEM SIMULATION IN
RESEARCH AND EDUCATION

INSIGHT—An interactive graphic instructional aid for systems analysis..	351	M. J. Merritt R. Sinclair
An interactive class-oriented dynamic graphic display system using a hybrid computer.....	357	A. Frank
Hybrid terminal system for simulation in science education.....	361	D. C. Martin
BIOMOD—An interactive computer graphics system for modeling.....	369	G. F. Groner R. L. Clark R. A. Bermam E. C. DeLand
The future of on-line continuous-system simulation.....	379	H. M. Aus G. A. Korn

COMPUTER STRUCTURES—PAST PRESENT AND FUTURE
(PANEL SESSION)

Position paper.....	387	C. G. Bell A. Newell
Position paper.....	395	F. P. Brooks, Jr.
Position paper.....	395	D. B. G. Edwards
Position paper.....	395	A. Kay

COMPUTERS IN SPORTS (PANEL SESSION)

Position paper.....	397	G. Brandt
Position paper.....	397	L. Eppele
Position paper.....	398	A. Lalchandani
Position paper.....	399	K. Mitchell
Position paper.....	399	K. G. Purdy
Position paper.....	400	F. B. Ryan

TWENTY YEARS IN PASSING (PANEL SESSION)

(No papers in this volume)

DATA SECURITY IN DATA BASE SYSTEMS

Multi-dimensional security programs for a generalized information retrieval system.....	571	J. M. Carroll R. Martin L. McHardy H. Moravec
for statistical purposes.....	579	M. H. Hansen
The formulary model for flexible privacy and access controls.....	587	L. J. Hoffman
Insuring confidentiality of individual records in data storage and retrieval		

THE APPLICATION OF COMPUTERS TO URBAN PLANNING AND DEVELOPMENT

Integrated municipal information systems: Benefits for cities--Requirements for vendors.....	603	S. E. Gottlieb
Geocoding techniques developed by the census use study.....	609	C. C. Smith M. S. White, Jr.
Urban COGO--A geographic-based land information system.....	619	B. Schumaker

SELECTED PAPERS IN DISCRETE SIMULATION

Understanding urban dynamics.....	631	G. O. Barney
Bankmod--An interactive decision aid for banks.....	639	W. P. Hoenhenwarter K. E. Reich
Simulation of large asynchronous logic circuits using an ambiguous gate model.....	651	S. G. Chappell
Adaptive memory trackers.....	663	G. Epstein

PLANNING COMMUNITY INFORMATION UTILITIES (PANEL SESSION)

Position paper.....	669	B. W. Boehm
Position paper.....	670	N. D. Cohen
Position paper.....	671	B. Nanus
Position paper.....	671	N. R. Nielsen
Position paper.....	672	E. B. Parker

COMPUTERS AND THE PROBLEMS OF SOCIETY (PANEL SESSION)

Position paper.....	675	P. Kamnitzer
Position paper.....	675	N. F. Kristy
Position paper.....	676	J. McLeod
Position paper.....	677	E. W. Paxson
Position paper.....	677	R. Weinberg

NUMERICAL METHODS

On the hybrid computer solution of partial differential equations with two spatial dimensions.....	401	G. A. Bekey M. T. Ung
Numerical solution of partial differential equations by associative processing	411	P. A. Gilmore
Consistency tests for elementary functions.....	419	A. C. R. Newbery

LABORATORY AUTOMATION

Laboratory automation at General Electric corporate research and development.....	423	P. R. Kennicott V. P. Scavullo J. S. Sicko E. F. Lifshin
Multicomputer processing in laboratory automation.....	435	C. E. Klopfenstein C. L. Wilkins
Enhancement of chemical measurement techniques by real-time computer interaction.....	441	S. P. Perone
The television/computer system—Acquisition and processing of cardiac catheterization data using a small computer.....	455	H. J. Covvey A. G. Adelman C. H. Felderhof P. Mendler E. D. Wigle K. W. Taylor
Cost benefits analysis in the design and evaluation of information systems..	469	I. Learnman
Factors to be considered in computerizing a clinical chemistry department of a large city hospital.....	477	R. L. Morey M. C. Adams E. Laga

DATA BASE SYSTEMS DESIGN

Integrated information system.....	491	J. C. Pendleton
A machine independent FORTRAN data management software system for scientific and engineering applications.....	501	I. Hirschsohn
Requirements for a generalized data base management system.....	515	A. C. Patterson IV

INTERACTIVE TEXT EDITING SYSTEMS

User engineering principles for interactive systems.....	523	W. J. Hansen
Computer assisted tracing of text evolution.....	533	W. D. Elliott A. Van Dam W. A. Potas

PLANNING AND DESIGNING OF HIGH PERFORMANCE SYSTEMS

Planning computer services for a complex environment.....	541	J. E. Austin
A high performance computing system for time critical applications.....	549	T. J. Gracon R. A. Nolby F. J. Sansom
Effective corporate networking, organization, and standardization.....	561	P. L. Peck

A universal cyclic division circuit

by ANDREW W. MAHOLICK and RICHARD B. FREEMAN

IBM Corporation
Research Triangle Park, North Carolina

INTRODUCTION

Recent innovations in circuit technology have allowed design alternatives that previously would have been economically unsound. LSI technology permits the use of generalized systems containing more logic than the specialized systems used in the past, implemented in unit logic, and at even lower cost. Five years ago an engineer would not have even considered using a cyclic redundancy checking circuit in the manner described here.

Cyclic Redundancy Checking (CRC) is a relatively old technique for use in error detection. W. W. Peterson and D. T. Brown¹ wrote a fundamental paper pointing out the great potentialities for cyclic codes in error detection and the requirements for implementing such error detection systems. The specialized serial case, i.e., with one input channel and one output channel, has been extensively studied and is contained in the Peterson² text. Many related papers, including pioneering efforts on this subject, are contained in a book edited by Kautz,³ Hsiao and Sih,⁴ Hsiao,⁵ and Patel⁶ have concentrated on the generalized case of CRC circuits with parallel multiple-channel inputs and outputs.

The above articles emphasize the use of fixed wiring patterns to implement the error-detection capabilities of cyclic redundancy codes. The hardware would require a complete rewiring to change the polynomial for the cyclic redundancy check. This, in turn, would mean that the circuitry itself would have a limited usefulness because only one type of polynomial could be used within a system at any one time.

Conventional CRC circuits for a given polynomial and data character size consist of a serial-by-bit shift register with EXCLUSIVE OR feedback circuits in those bit positions which represent a term in the CRC polynomial. Figure 1 shows an implementation for the polynomial, $x^{16} + x^{15} + x^2 + 1$. In a digital data communication system, this bit-synchronous scheme must usually be duplicated for each communication line. In

communication line multiplexers the logic is sometimes shared, but it is limited to those communication lines using the same CRC checking polynomial.

We shall describe a generalized method for updating cyclic redundancy checking logic at the character level which is capable of operating upon any data character size in conjunction with any checking polynomial of a given length.

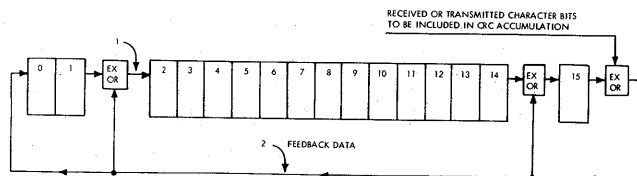


FIGURE 1 - SIMPLIFIED SERIAL IMPLEMENTATION OF THE POLYNOMIAL, $x^{16} + x^{15} + x^2 + 1$

Figure 1—Simplified serial implementation of the polynomial $X^{16} + X^{15} + X^2 + 1$

We shall describe the device from the point of view of its application in a digital data communication line multiplexer where a variety of CRC polynomials might be employed to service multiple communication lines. However, it should be noted that this device can be used by future terminals as well as by the increasing number of I/O devices (tapes, discs, et al.) which are employing CRC checking.

THE EVOLUTION FROM SERIAL TO PARALLEL

In this section we shall trace the evolution of the polynomial, $X^8 + X^5 + X^3 + X + 1$, from its serial-by-bit implementation as shown in Figure 2 to its parallel-by-character implementation as shown in Figure 7. This will provide the background necessary to understand

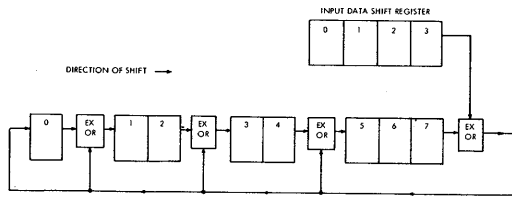


Figure 2—Simplified serial implementation of the polynomial $X^8 + X^5 + X^3 + X + 1$

the generalized method described later that can process f bits in parallel for any arbitrary checking polynomial to generate the CRC character or the syndrome.

From Patel,⁶ we have the required theoretical relationship between the serial-by-bit and parallel-by-character cases. A brief section is reproduced for the convenience of the reader in Appendix A.

Figure 2 shows the conventional serial-by-bit implementation for the polynomial, $X^8 + X^5 + X^3 + X + 1$, to be used in conjunction with a four-bit data character.

Figure 3 shows an equivalent circuit in which redundant EXCLUSIVE-OR circuits have been added, such that there is one at the input of each stage of the shift register. Those EXCLUSIVE-ORS not required to implement the polynomial have one input connected to a "logical 0" voltage level. Thus, the input coming from a previous shift register stage will pass through as if the EXCLUSIVE-OR circuit were not there, i.e., $0 + X = X$.

Figure 4 shows another functionally equivalent circuit. Some flexibility is obtained by the addition of an AND circuit, which controls one input of the EXCLUSIVE-OR circuit. One input of each AND circuit is connected in common to the feedback path. The other input of each AND circuit may be connected to either a logical 1 or 0 voltage level according to whether or not the corresponding term exists in the polynomial.

Figure 5 shows another step in the evolutionary process. The data is entered parallel-by-character rather

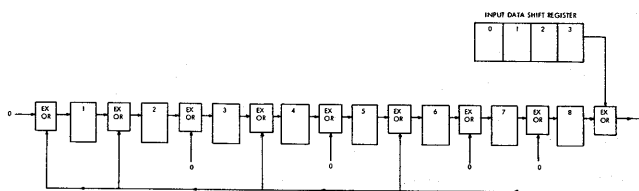


Figure 3—Equivalent implementation #1 for the polynomial $X^8 + X^5 + X^3 + X + 1$

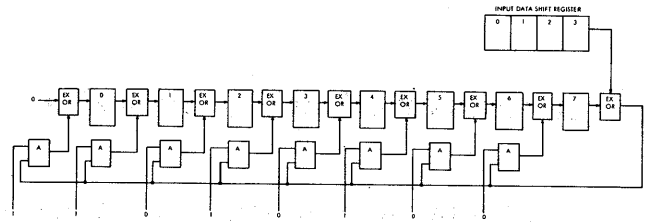


Figure 4—Equivalent implementation #2 for the polynomial $X^8 + X^5 + X^3 + X + 1$

than serial-by-bit. This is accomplished by EXCLUSIVE-ORing the data character with the corresponding low order bits of the shift register prior to shifting. This may be done since the contents of the shift register will be the same after f -bit shifts on a serial-by-bit basis or if the f -bits are EXCLUSIVE-ORed with the low order stages of the shift register and then allowing the f -bit shifts to occur.

Figure 6 shows the next step in the evolutionary process. Here one row of shift registers has been added for each data bit. For all rows except the first, the input of an EXCLUSIVE-OR circuit in cell position $C_{n,m}$ is connected to the output of cell $C_{n-1, m-1}$, where n is the row number and m is the column number. It is connected there rather than to the cell on its immediate left, $C_{n, m-1}$. We shall shift each row from 1 to 4 on a mutually exclusive basis. When the last row has been shifted, the output of row 4 will be identical to the serial-by-bit implementation after four bit shifts.

In the final equivalent circuit every shift register state is deleted such that all that remains is the combinational logic as shown in Figure 7. In this version, the only time delay that will be encountered is the propagation delay of the logic elements.

We still need some memory elements, however. We require an OLD CRC REGISTER, a NEW CHARACTER REGISTER, and a NEW CRC REGISTER. The NEW CHARACTER REGISTER and the low

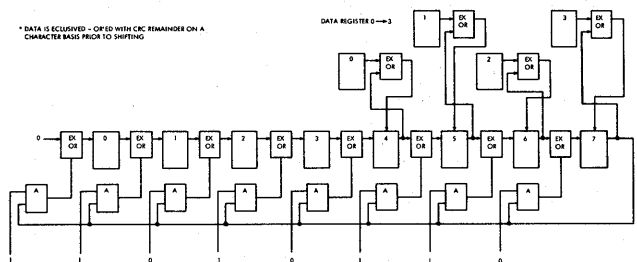


Figure 5—Equivalent implementation #3 for the polynomial $X^8 + X^5 + X^3 + X + 1$

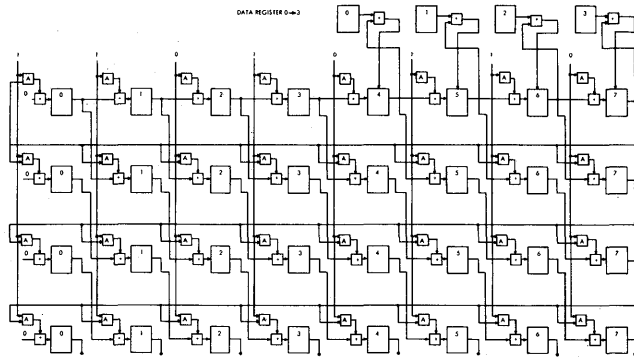


Figure 6—Equivalent implementation #4 for the polynomial $X^8+X^5+X^3+X+1$

order position of the OLD CRC REGISTER are EXCLUSIVE-ORed together. The outputs of the exclusive or circuits plus the high order positions of the OLD CRC REGISTER are connected to appropriate positions in the first row of the array. The updated CRC remainder will appear at the output of the bottom row and will be set in the NEW CRC REGISTER. The contents of the NEW CRC REGISTER can then be transferred to the OLD CRC REGISTER in preparation for the next iteration.

A POLYNOMIAL REGISTER set to the required bit configuration is used in lieu of fixed wiring to select the polynomial. It offers more than is required for the single polynomial implemented since it will provide for any polynomial of the eighth degree.

THE UNIVERSAL CRC REGISTER

For a practical implementation in a communication multiplexer, the system (Figure 8) uses a memory de-

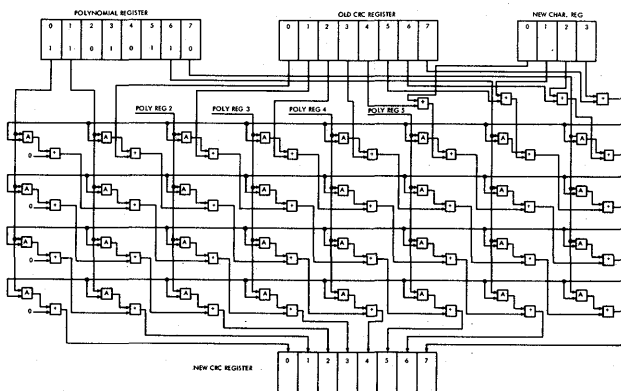


Figure 7—Equivalent implementation #5 for the polynomial $X^8+X^5+X^3+X+1$

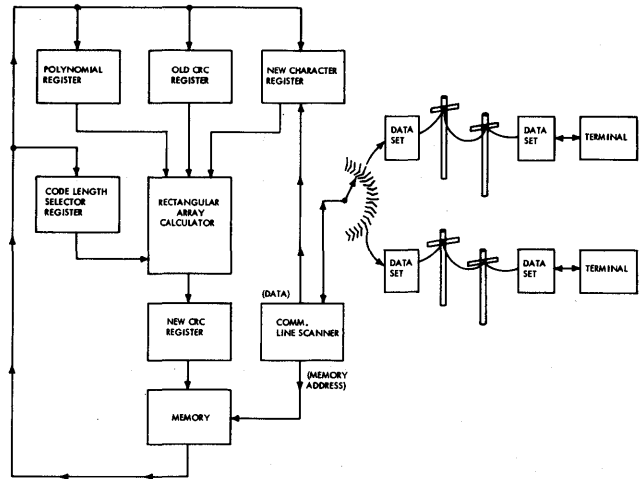


Figure 8—Functional block diagram of universal CRC logic

vice which is addressable via a communication line scanner. The location accessed in the memory for a particular line contains unique line control information including the current CRC value, data character length, and a binary representation of the polynomial associated with that line. Subsequent to the receipt of the transmission line address, the memory will be accessed to obtain specific parameters associated with that address to set the CODE LENGTH SELECTOR (6, 7, or 8 bits), POLYNOMIAL, and OLD CRC registers. The old CRC is the cyclic redundancy check remainder calculated for the previous data characters received or transmitted during the current transmission on the line.

At the same time that the transmission line address is made available to memory, the new data character to be serviced from this line is stored in the NEW CHARACTER register.

When all the parameters associated with the transmission line address have been set, the CODE LENGTH SELECTOR, POLYNOMIAL, OLD CRC and NEW CHARACTER registers are gated to the inputs of an array calculator.

The array calculator is an asynchronous device which will continually calculate a cyclic redundancy check upon the data contained within the POLYNOMIAL register, the OLD CRC register, the NEW CHARACTER register, and the CODE LENGTH SELECTOR register. The output of the array calculator, after a sufficient amount of propagation delay time within the array calculator, is the new CRC value and it is stored in the NEW CRC register. The new CRC contained in the NEW CRC register is then stored in memory at the same location as the old CRC

was previously stored. On the next iteration, this data will be the old CRC remainder.

CRC calculation continues in a multiplexed fashion. Each communication adapter invokes the CRC parameters associated with it by presenting a unique memory address to access the memory. This insures that the proper old CRC, code length, and polynomial are combined with the new data character to generate the new cyclic redundancy check remainder.

An alternate approach will be to transmit the cyclic redundancy check following the stop character and allow the cyclic redundancy check and the stop character to pass through the universal cyclic redundancy check generator. The result of this operation would be a data word as an output from the array calculator which represents the syndrome. A non-zero syndrome indicates an error in the received data. However, if the syndrome is zero we know only that the received bit stream is one of the allowable set of transmitted bit streams. It may not be the actual transmitted bit stream. That is, an undetectable error may have occurred.

OPERATIONAL CHARACTERISTICS

Figure 9 is a detailed presentation of the input logic of the rectangular array calculator which provides for 6-, 7-, or 8-bit data characters and polynomials of order 16 or less. The POLYNOMIAL and OLD CRC registers are 16-bit registers labeled from 0 to 15 and the NEW CHARACTER register is an 8-bit register labeled 0 to 7. Only the first two rows of the rectangular array are shown in Figure 9.

Each of the registers is always filled from data busses entering these registers such that the right-most binary-bit positions in each of these registers represent data corresponding to the particular polynomial terms, the

old CRC value, and the new data character which is required to update the CRC value. In cases where this data does not fill the entire register, the higher order or left-most bit positions are forced to a binary 0 condition as is shown above each of the registers in Figure 9.

While the logic for the array shown in Figure 9 looks very extensive, it should be noted that this is deceptive, since the logic has intentionally been designed as an iterative structure to make it attractive for large scale integration (LSI). The array could be packaged on one chip, making the large amount of logic involved of little consequence.

The logic shown in Figure 9 performs a relatively complicated mathematical function upon the various register inputs to the array, initially, a modulo-two addition (half summing) occurs between the old CRC and new data character. The result of that addition is then applied to the array calculator. The array calculator operates in a manner so as to duplicate mathematically the results which might be obtained by serial feedback approaches to CRC generation as previously shown.

The circuitry within the array has its various analogies to serial feedback shift register implementation. For example, the vertical lines such as line 1 in Figure 9 represents a single feedback point in an analogous serial feedback approach to CRC generation (Line 1 in Figure 1). The vertical line represents the presence ("1") or absence ("0") of a term in the chosen cyclic check polynomial. For instance, the polynomial $x^{16} + x^{15} + x^2 + 1$ used in Binary Synchronous Communication would be represented with "1's" in positions 15, 2, and 0 ($1 = x^0$) of the POLYNOMIAL register. In effect, there is always a high order term (16 in this case) which necessitates the initial modulo-two addition.

To determine the right-justified positions in the POLYNOMIAL register for polynomials of degree less than 16, it is necessary to multiply the polynomial by x raised to the power (16-(degree of polynomial)). For instance, the polynomial $x^6 + x^5 + 1$ would be implemented as though it were $x^{(16-6)}(x^6 + x^5 + 1) = x^{16} + x^{15} + x^{10}$ and "ones" would be placed in positions 10 and 15 with position 16 implied. LRC for 8 bit codes ($x^8 + 1$) would be implemented as $x^{(16-8)}(x^8 + 1) = x^{16} + x^8$ with a "one" in position 8.

The horizontal line or intermediate feedback line, such as line 2 of Figure 9, represents for each bit shift the state of the feedback network in the serial feedback approach to CRC generation (Line 2 of Figure 1). The horizontal line is always the output of the right-most position in the row above in the rectangular array. The concurrence of a feedback path and the proper data bit in the feedback path would cause a change in the

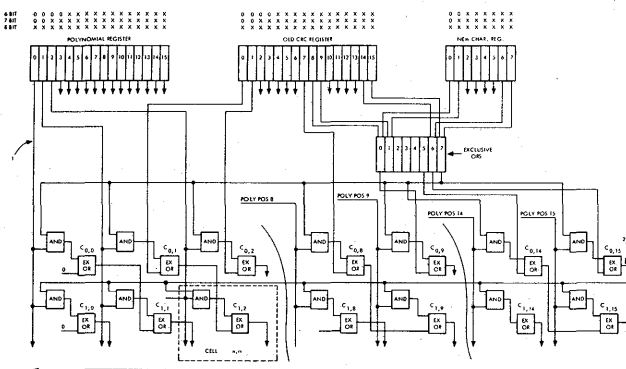


Figure 9—Input logic for the rectangular array

data within the serial shifting network. A similar changing of data occurs in the transmission between one cell element and another if the data on the intermediate feedback line and the line from the polynomial register are of the proper values. The output ("1" or "0") of a given cell is equal to the output of the cell diagonally above and to the left of it unless it is reversed by the coincidence of a logical "one" on both the vertical line, and on the horizontal line associated with the position.

THE ARRAY AND ITS OPERATION

Certain machines might interface the array in a different fashion than shown in Figure 9, i.e., the CRC polynomial select register might be replaced by permanent wiring in a terminal application, or the output assembler (described later) might be reduced or eliminated if only one data character length exists.

The array consists of replicas of the simple circuit shown in Figure 10. The cell is shown enclosed in the dotted line labeled cell $C_{n,m}$ on Figure 9. Each cell element has three inputs. The first, 1, is connected to a line 4 carrying signals representing the binary value for the intermediate feedback within the array calculator. The second input, 2, is connected to a line 5 which has

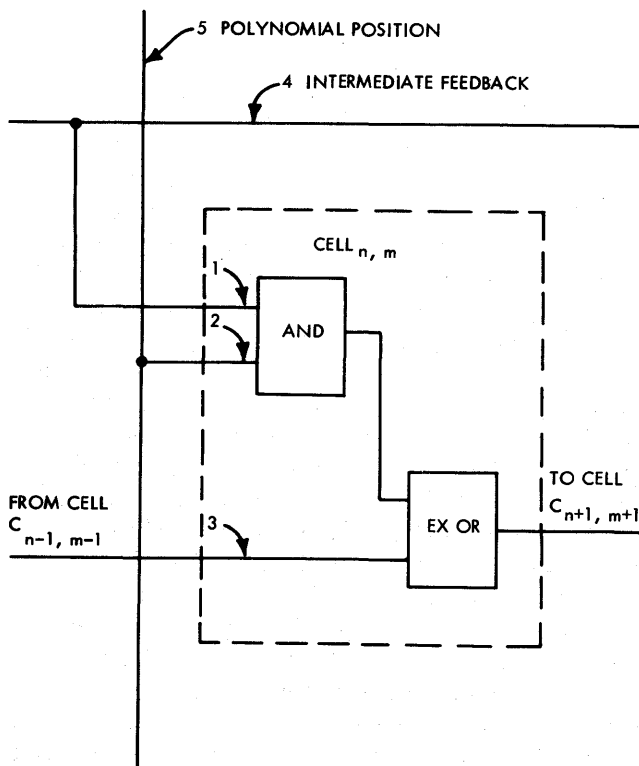


Figure 10—Logic diagram for a standard cell of the array

binary information representing the binary value of a given single bit position within the polynomial, and is connected directly to the POLYNOMIAL REGISTER. A third input, 3, is a connection to a cell which is diagonally upward to the left of the array. Specifically, Cell $C_{n,m}$ has its third input connected to the output of cell $C_{n-1,m-1}$, where n designates the row number and m the column number.

For the cells in the leftmost column, there are no positions diagonally upward to the left. Therefore, the third input to the cell is wired permanently to a voltage source having a binary value of 0.

The cell elements along the first row of the array have a slightly different characteristic than the other cells of the array because the third input to each cell cannot be connected to the cell element diagonally upward to the left within the array since no such element exists for those in the first row. For cell element $C_{0,0}$, cell element row 0 and column 0, the third input is wired to a binary 0 voltage level. For cell element $C_{0,1}$, the cell element in row 0 and column 1, the third input is connected to bit position 0 of the OLD CRC register. Subsequent cells in row 0 have their third input connected directly to the OLD CRC register up to and including cell $C_{0,7}$.

For cells $C_{0,8}$ to cell $C_{0,15}$, the third input to each cell is wired in a different manner than for the other cells within the row. Cell $C_{0,15}$ provides a good example. The third input to this cell is connected to EXCLUSIVE OR circuit 6. The inputs to EXCLUSIVE OR 6 are connected to bit position 6 of the NEW CHARACTER register and to bit position 14 of the OLD CRC register. Similar wiring exists for the other array elements $C_{0,8}$ through $C_{0,14}$.

The intermediate feedback signal from EXCLUSIVE OR 7 is connected to the first input to each of the cell elements in row 0. The intermediate feedback signal is generated by EXCLUSIVE OR circuit 7. The inputs to EXCLUSIVE OR circuit 7 are connected to bit position 7 of the NEW CHARACTER register and to bit position 15 of the OLD CRC register.

PROVISION FOR VARIOUS CODE LENGTHS

Although the concept is general enough to accommodate other code lengths, it is assumed that 6-, 7-, and 8-bit codes may be used and that the polynomials associated with these code lengths can be of degree 6 or 12, 7 or 14, and 8 and 16, respectively. The same array may be used to accomplish this by "right justifying" it. For example, a 7-bit code of polynomial degree 14, would extend from 2 to 15 in the POLYNOMIAL register. Positions 0 and 1 would be set to zero. A 7-bit

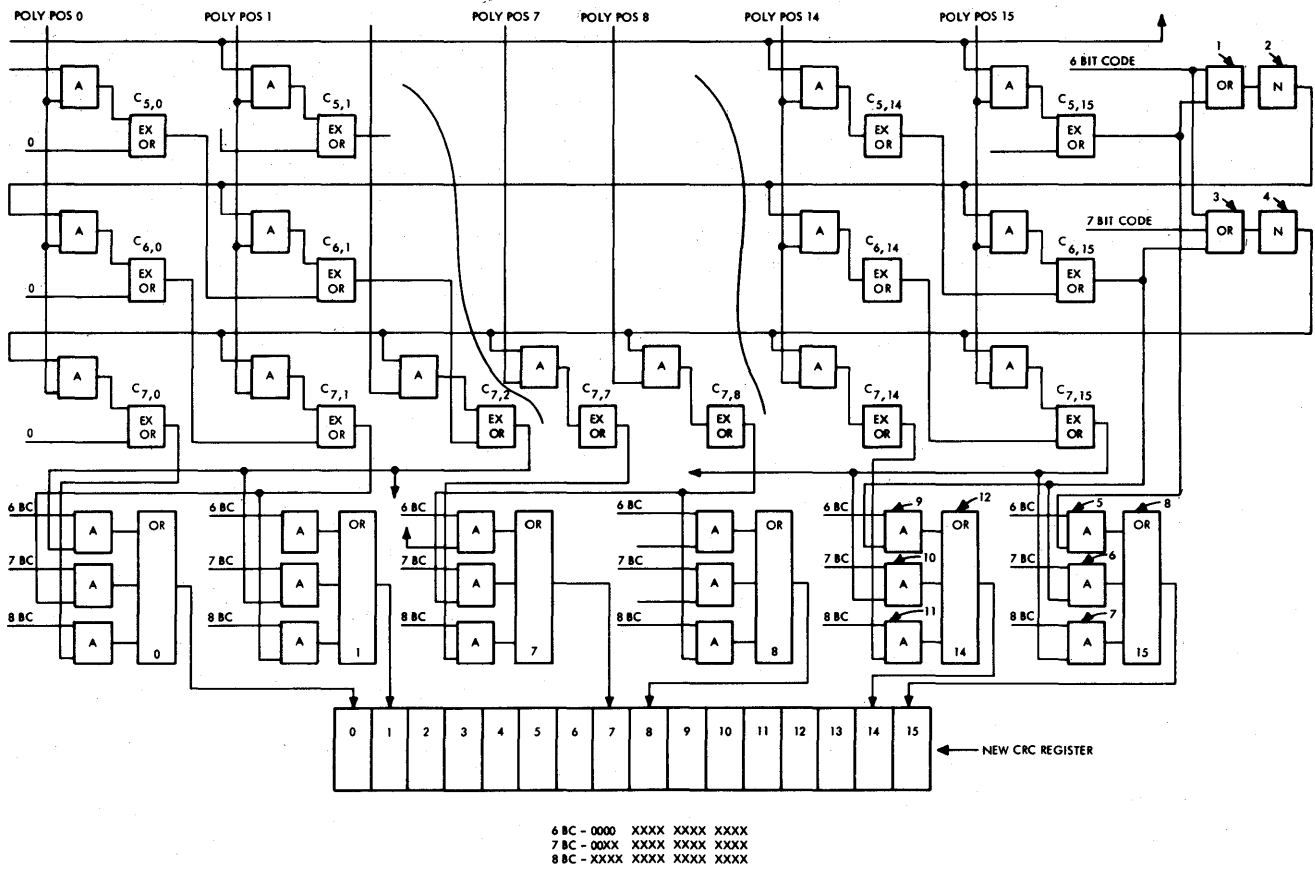


Figure 11—Output logic for the array

code of degree 7 would extend from positions 9 to 15 in the POLYNOMIAL register. Positions 0 to 8 would be set to zero. This method appropriately truncates the “width” of the array.

However, the “depth” of the array must also be truncated in accordance with the character length. Each row of the array represents a serial shift of one bit. Thus, for a 6-bit code, using an array designed for eight bits, the desired answer is present and properly aligned at the outputs of the sixth row. However, because of chip layout limitations, it is not practical to bring out independent outputs from each of several rows.

Thus, a compromise is struck by degating the intermediate feedback path to lower rows, which results in a single right shift of the answer for each such row. Note that the right-most bits wrap around the bottom right side, appearing as the outputs of the right-hand positions of the second row up (for a 7-bit code) or subsequent rows for shorter codes. Thus for multi-length systems it will be necessary to assemble for proper alignment some time before the NEW CRC becomes the next OLD CRC. See Figure 11. The resultant

alignment is as shown at the bottom of Figure 11 for 6-, 7-, and 8-bit code lengths and 12-, 14-, and 16-degree polynomials, respectively.

The output of the array calculator must be taken from the proper cells within the array and this is dependent upon the particular bit length of the character for which the cyclic redundancy check is being calculated. For example, should the character upon which the CRC is being calculated be of a length of only six bit positions, the output should be taken from the output of row number 5, (the first row being identified by a 0). This is accomplished through various circuit elements within the array calculator as shown in Figure 11. Specifically, OR circuits 1 and 3 are activated by a signal indicating that the new character is of a 6-bit code type. The outputs of the OR circuits are inverted and then propagated along the intermediate feedback signal paths to disable the AND circuits in each of the cell elements in rows 6 and 7. As a consequence, the cell elements in rows 6 and 7 will not modify the data received from the outputs of the cell elements within row number 5, and they can be used to propagate the output from the cell elements in row 5.

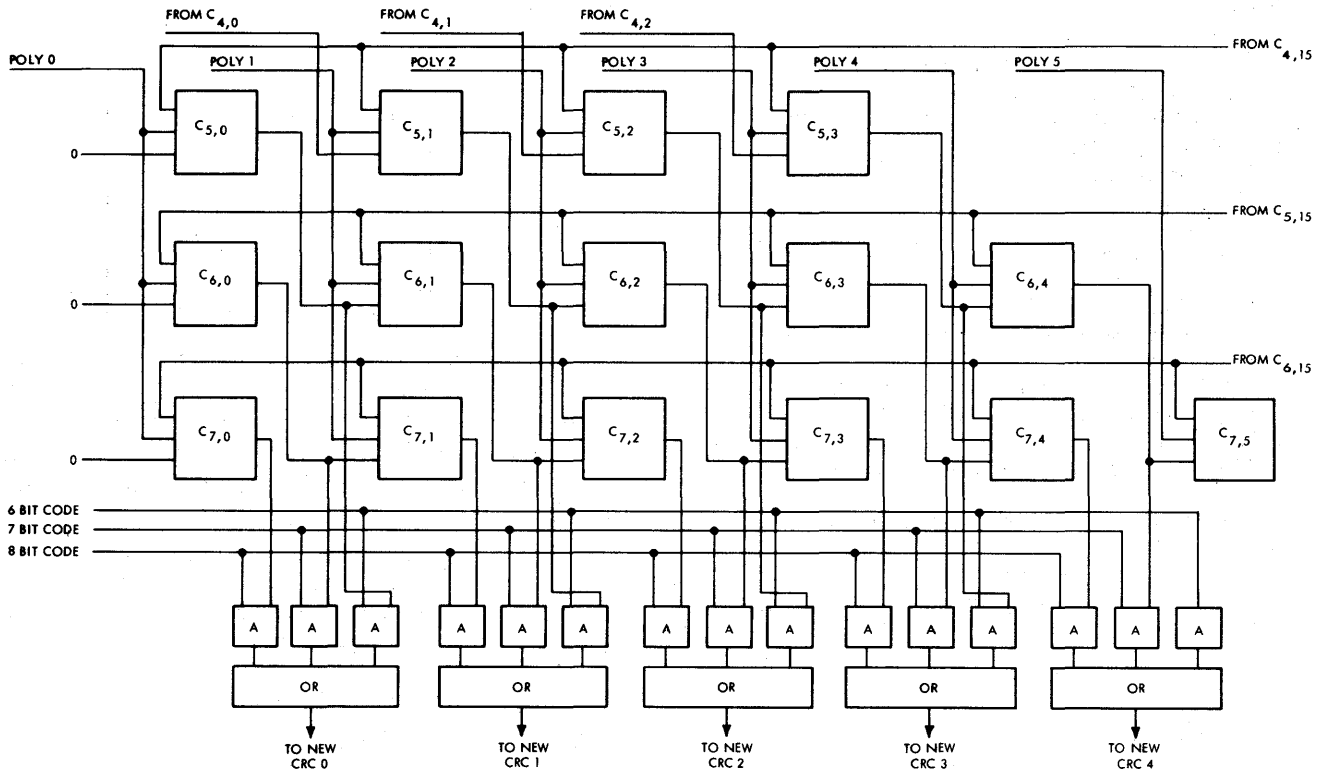


Figure 12—Alternate output logic for the array

The output of Cell $C_{5,15}$ is propagated to AND circuit 5. When a 6-bit code is selected, a positive voltage will appear on the second input to AND circuit 5. The data appearing on the output of cell $C_{5,15}$ would then be transmitted via AND circuit 5 or OR circuit 8 and on to bit position 15 of the NEW CRC register.

Bit position 14 of the output is gated from cell element $C_{6,15}$ to AND circuit 9 when a 6-bit code is indicated. AND circuit 9 has an output connected to OR circuit 12 whose output is connected to bit position 14 of the NEW CRC register.

Cell element $C_{7,15}$ provides the output for bit position 13 of the NEW CRC register when a 6-bit code is being operated upon. This is accomplished by gating circuitry not shown. The other bit positions of the NEW CRC REGISTER would be filled from data from cell elements in row 7 of the array in a similar manner to that described for bit position 13 in the NEW CRC REGISTER when a 6-bit code was being transmitted. In the case where the new character contains eight data bits, each of the outputs of the eighth row of the array calculator would be connected directly to the NEW CRC register via appropriate switching circuits and no compensation for the shift in the array network would be necessary.

The gating circuitry above-mentioned in connection

with Figure 11, is particularly adapted to LSI circuitry because the output gating occurs from elements of the network which are on the peripheries of the rectangular array calculator. With the above scheme, the array could easily be placed in a single chip and all wiring connections can be made to points within the array without crossing any internal connections.

The advantage to the above-shown output gating is that additional wires from the outside of the array are not necessary to connect to interior points within the array. Where such wiring problems do not exist, a simpler approach to the outputting is shown in Figure 12. This logic is a simple AND-OR assembler where row 5, 6, or 7 is selected for gating into the NEW CRC register, depending on whether the code length is 6, 7, or 8 bits, respectively. This assembly function can be considered as part of the array element and can therefore be extended throughout the array to provide for any code length.

ACKNOWLEDGMENT

The authors wish to acknowledge the hardware implementation contribution of M. T. Kawalec and S. R. Stager, III.

REFERENCES

- 1 W W PETERSON D T BROWN
Cyclic codes for error detection
Proceedings of the IRE pp 228-235 January 1961
- 2 W W PETERSON
Error correcting codes
MIT Press 1961
- 3 W H KAUTZ
Linear sequential switching circuits
Holden-Day Inc 1965
- 4 M Y HSIAO K Y SIH
Serial-to-parallel transformations of feedback shift register circuits
IEEE Transactions on Electronic Computers
VOL EC-13 pp 738-740 December 1964
- 5 M Y HSIAO
Theories and applications of parallel linear feedback shift register
IBM TR 1708 SDD Poughkeepsie March 1968
- 6 A M PATEL
A multi-channel CRC register
AFIPS Conference Proceedings Vol 38 pp 11-14 Spring 1971

APPENDIX A

The following is reproduced from Patel⁶:

In this section, we develop the mathematics for obtaining a multi-channel CRC register that can process f bits in parallel to generate the CRC character or the syndrome. One shift in the parallel circuit is equivalent to f shifts in the corresponding serial CRC register. The number f is a positive integer, smaller than the degree r of the checking polynomial.

$G(x)$ denotes the checking polynomial, often called the generator polynomial. We use the following notation:

$$G(x) = G_0 + G_1x + G_2x^2 + \dots + G_rx^r \quad (1)$$

The state vector $X_t = (x_0, x_1, \dots, x_{r-1})t$ denotes the contents of the CRC register at time t . T denotes the companion matrix of the polynomial $G(x)$, corresponding to the serial CRC register connections. Let z_t denote the data bit entering the serial CRC register at time t . Then the shifting operation of the serial CRC register is given by the (mod-2) matrix equation

$$X_{t+1} = X_tT \oplus z_tG \quad (2)$$

where G is the vector $(G_0, G_1, G_2 \dots G_{r-1})$, and T is

given by:

$$T = \begin{bmatrix} 0 & 1 & & & \\ & 0 & 1 & & \\ & & & \ddots & \\ & & & & 1 \\ G_0 & G_1 & G_2 & \dots & G_{r-1} \end{bmatrix} \quad (3)$$

Suppose that $z_t, z_{t+1}, \dots, z_{t+f-1}$ are the f data bits (a byte) entering successively into the serial CRC register during the f consecutive shifting operations. The contents of the CRC register at the end of f shifts is denoted by the vector X_{t+f} . Using Equation 2 iteratively, f times, one can obtain:

$$X_{t+f} = X_tT^f \oplus z_tGT^{f-1} \oplus z_{t+1}GT^{f-2} \oplus \dots \oplus z_{t+f-1}G \quad (4)$$

Here T^j is the j th power of the matrix T . Let Z_t denote the input data sequence, as follows:

$$Z_t = (z_{t+f-1}, z_{t+f-2}, \dots, z_{t+1}, z_t)$$

Let D denote the following partitioned matrix:

$$D = \begin{bmatrix} G \\ \hline GT \\ \hline GT^2 \\ \hline \hline \hline \\ \hline GT^{f-1} \end{bmatrix} \quad (5)$$

Note that the vectors $G, GT, GT^2, \dots, GT^{f-1}$ represent the contents of the serial CRC register as the vector G is shifted $f-1$ times.

Then, Equation 4 can be rewritten as:

$$X_{t+f} = X_tT^f \oplus Z_tD \quad (6)$$

The sequential circuit realizing Equation 6 has the property that with the input byte Z_t (f bits in parallel), it changes from state X_t to X_{t+f} in a single shift. This is the equivalent operation to f shifts of the corresponding serial CRC register with the same input data entered serially.

Cyclic redundancy checking by program

by P. E. BOUDREAU and R. F. STEEN

IBM Corporation

Research Triangle Park, N.C.

INTRODUCTION

Recent advances in the use of mini-computers as control elements of a computer complex and as intelligent terminals¹ are indicative of a trend toward relocation of certain hardware functions to micro-program or machine level program. One such function which is a particularly good candidate, for various reasons, has already been moved into program in several machines (e.g., IBM System 360/25 Integrated Communication Adapter² and the IBM 1130³). This function is error control using an error detection Cyclic Redundancy Check (CRC). A CRC is a variable length shortened cyclic code in which a message is a code word if, and only if, the message polynomial $M(x)$ is divisible by the generator polynomial $G(x)$.

Error detection and correction codes have been studied extensively for more than 15 years. The most comprehensive references,^{4,5} as well as the majority of papers written in the area, measure the encoding and decoding complexity in terms of the cost of hardware and the time for decoding. With some notable exceptions,^{6,7} very little attention is given to the problem of encoding and decoding using machine level or micro-instructions. However, in some cases such as the Berlekamp algorithm³ for BCH codes, it may very possibly be easier to write a program for certain steps of the decoding procedure than to design hardware. Programmed error correction is especially appealing for use with high rate codes when error probabilities are low, since, in this case, a major portion of the correction process need only be performed when errors actually occur. Allocation of a significant amount of hardware for these relatively infrequent events is expensive. Furthermore, rapidly advancing memory technology helps to make program-controlled devices not only economically feasible but attractive.

One part of the problem is addressed in this paper. It is the problem of encoding or generating check bits. The solution, however, also applies to the decoding problem for error detection codes of this type. A similar approach, based on the properties of the companion matrix, has been used for parallel hardware devices.^{8,9} With this approach, efficient and attractive programs can be developed for software or firmware. Subroutines developed here require as few as six instructions with sequential instruction execution to update a 16-bit remainder for eight new information bits. A program directly simulating a shift register would require at least three instructions (EXCLUSIVE OR, SHIFT, and BRANCH) per bit, or 24 instructions for an eight-bit update.

MATRIX APPROACH TO CYCLIC CODES

In this section, we review the relationship between multiplication by the companion matrix and polynomial division used to generate a code word. We then generalize the operation to an m -bit character-by-character operation developing a matrix equation to update the calculated redundancy m bits at a time. The appendix will be helpful to those familiar with the shift register in order to further justify the connection between the shift register operation and the matrix multiplication.

Generally, the check bit generation process is one of determining $R(x) = x^h I(x) \bmod G(x)$ where $I(x)$ is the polynomial whose coefficients are the information bits and h is the number of check bits. We can next let the coefficients of $R(x)$ be an h bit vector, R , and let G be the h by h companion matrix shown below. The binary digits, g_i , $i = 1, 2, 3 \dots h-1$, are the coefficients of the generator polynomial.

$$G = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ & & & \cdot & \\ & & & \cdot & \\ & & & & \cdot \\ 0 & 0 & 0 & \dots & 1 \\ 1 & g_1 & g_2 & \dots & g_{h-1} \end{bmatrix}$$

Then, if we let $b(1) = i_{k-1}$ be the first information bit (the $k-1$ th coefficient of $I(x)$) and $b(k) = i_0$ be the last information bit, it is clear (see the appendix or Reference 7) that the remainder R can be calculated iteratively using the following formula:

$$A(t+1) = \{A(t) + [0, 0, \dots, 0, b(t+1)]\} \cdot G \quad (1)$$

and setting $R = A(k)$. It should be noted that $A(t)$ represents the remainder of $x^t I_t(x)$ divided by $G(x)$ which is the calculated redundancy after the first t information bits, $I_i(x)$, have been taken into account.

We now define $B(t+1) = [0, 0, \dots, 0, b(t+1)]$ and rewrite Equation (1 or A2) as

$$A(t+1) = [A(t) + B(t+1)]G. \quad (2)$$

Equation (2) is the basic matrix description of the polynomial division process (circuit function) on a bit-by-bit basis. The advantage of the matrix approach is realized when one extends it to a multibit or character level. We can do this for m bits-per-character as follows, assuming $m \leq h$. Repeated use of Equation (2) yields:

$$\begin{aligned} A(t+m) &= [A(t+m-1) + B(t+m)] \cdot G \\ &= \{[A(t+m-2) + B(t+m-1)] \cdot G \\ &\quad + B(t+m)\} \cdot G \\ &\quad \vdots \\ &= A(t) \cdot G^m + \sum_{j=1}^m B(t+j) \cdot G^{m-j+1}. \end{aligned} \quad (3)$$

Equation (3) expresses the remainder at time $t+m$ in terms of the remainder at time t and the next m input bits $b(t+1), b(t+2), \dots, b(t+m)$. This equation can be put into a better form by using the "shifting" property of the companion matrix G .

$$\begin{aligned} A(t+m) &= A(t) \cdot G^m \\ &+ [0, 0, \dots, 0, b(t+m), b(t+m-1), \dots, b(t+1)] \cdot G^m. \end{aligned} \quad (4)$$

If indeed we are operating with m bits per character and $A(t)$ is the remainder after some character has been sent, then $A(t+m)$, given by Equation (4), is the remainder after the next character has been sent and $b(t+m), b(t+m-1), \dots, b(t+1)$ is the bit string of length m representing that next character, where $b(t+1)$ is the first bit sent.

Since we will be using this from now on, it is convenient to make a slight change of notation. We define

$$A_j = [a_{0,j}, a_{1,j}, \dots, a_{h-1,j}]$$

as the remainder after the j th character, and

$$C_j = [0, 0, 0, \dots, 0, c_{0,j}, c_{1,j}, \dots, c_{m-1,j}]$$

as an h component vector where

$$c_{0,j}, c_{1,j}, c_{2,j}, \dots, c_{m-1,j}$$

is the bit string of length m representing the j th character and $c_{m-1,j}$ is the first bit of the character transmitted. That is

$$a_{i,j} = a_i(t) \quad \text{for } i = 0, 1, \dots, h-1$$

and

$$\begin{aligned} c_{0,j} &= b(t+m) \\ c_{1,j} &= b(t+m-1) \\ &\quad \vdots \\ c_{m-1,j} &= b(t+1). \end{aligned}$$

With this notation Equation (5) becomes the character-by-character version of Equation (2)

$$A_{j+1} = [A_j + C_{j+1}] \cdot G^m. \quad (5)$$

This equation expresses the remainder after $j+1$ characters as a function of the remainder after j characters and the $j+1$ st character for $m \leq h$ bits per character. It is the fundamental result which we apply below.

MATRIX IMPLEMENTATION OF CYCLIC CODES

This matrix description of cyclic checking leads directly and intuitively to several different programmed checking implementations. It is this feature which makes the approach valuable. Since instruction sets, core availability, and instruction execution times vary widely, three approaches will be described.

It is very convenient to describe these subroutines in APL¹⁰ with a single line of APL representing a single machine language instruction. For those interested in the exact operation of the simulated machine language instruction, a knowledge of basic APL is required; otherwise, the marginal machine instructions and

comments should clearly indicate the general nature of the operation on each line of code. It is assumed that there are four 16-bit registers which are available to the programmer. These are represented by the APL vector variables RA, RB, and RC with the fourth being the base register which is used for the return branch to the main program. In APL, RA[1;] represents the high order byte of register RA and RA[2;] represents the low-order byte of the same register. The storage area for tables is represented by the matrix SA which is as large as necessary.

Although we have assumed a 16-bit data path for the three examples, it is easy to write similar subroutines for an eight-bit ALU by partitioning the G^8 matrix in a different manner. We will use the terms, "byte" and "halfword" to mean eight and 16 bits respectively.

In general, our methods below are iterative schemes for finding the remainder using the recurrence relationship

$$A_{j+1} = [A_j + C_{j+1}]G^m.$$

For simplicity we define what we call a "working remainder" W_{j+1} ,

$$\begin{aligned} W_{j+1} &= [A_j + C_{j+1}] \\ &= [a_{0,j}, \dots, a_{h-m-1,j}, (a_{h-m,j} \oplus c_{0,j+1}), \\ &\quad \dots, (a_{h-1,j} \oplus c_{m-1,j+1})] \\ &= [w_{0,j+1}, w_{1,j+1}, \dots, w_{h-1,j+1}] \end{aligned}$$

Basically, our problem is to find A_{j+1} given W_{j+1} and G^m using

$$A_{j+1} = W_{j+1}G^m.$$

Since W_{j+1} is a binary vector of length h , it can take no more than 2^h values. The following methods, called the "one-256-halfword-table look-up," the "two-32-halfword-table look-up," and the "binary summation" method, are various ways to perform this job.

Purely for ease of notation, we now fix the values of h and m . We will let the number of parity bits be 16 ($h=16$) and the number of bits per character be eight ($m=8$). Substitution in (5) gives us the fundamental equation

$$A_{j+1} = W_{j+1}G^8 \quad (6)$$

where

$$\begin{aligned} W_{j+1} &= A_j + C_{j+1} \\ A_0 &= [0, 0, 0, \dots, 0, 0] \\ A_j &= [a_{0,j}, a_{1,j}, \dots, a_{15,j}] \\ C_j &= [0, 0, \dots, 0, c_{0,j}, c_{1,j}, \dots, c_{7,j}] \end{aligned}$$

are all 16-bit vectors, and

G^8 = the companion matrix raised to the 8th power.

We note again for emphasis that $c_{7,j}$ is the first bit of the j th character while the $j=1$ st character is the first character transmitted or received.

One-256-halfword-table look-up method

This is a simple one-table look-up method which requires a significant amount of storage and frequently will be impractical for codes with more than eight bits-per-character. However, it embodies most of the basic ideas of the matrix approach and is a good starting place. In an instruction set with the logical EXCLUSIVE OR operation, the forming of W_{j+1} is trivial. The next step is to find A_{j+1} which can be found by multiplying W_{j+1} by G^8 . This can be done very rapidly by table look-up. Rather than blindly storing all 2^{16} halfwords which can result from this operation, we notice that G^8 has the form

$$G^8 = \begin{bmatrix} 0 & | & I \\ \hline & & X \end{bmatrix}.$$

Thus $W_{j+1}G^8$ can be written

$$W_{j+1}^{(L)}X \oplus W_{j+1}^{(H)}[0 \mid I]$$

where $W_{j+1}^{(H)}$ is an eight-bit vector comprising the high-order eight bits of W_{j+1} and $W_{j+1}^{(L)}$ represents the low-order eight bits of W_{j+1} . If byte operations are available, the product $W_{j+1}^{(H)} \cdot [0 \mid I]$ is simply moving the byte from the high-order half of a 16-bit register to the low-order half. The second instruction in Table I performs this operation. The second product above requires a table look-up for one of 256 halfwords representing all possible values of $W_{j+1}^{(L)} \cdot X$. This is done in instruction four after the program has shifted the address left one bit in order to force the address to a halfword boundary. The table is assumed to be located on a 512 byte boundary. Its address is stored in the seven low-order bits of the high-order byte of the RB register. The two results are EXCLUSIVE Ored together in the fifth instruction and the table address is restored in the last instruction before the return branch. Table I shows the program which will update the CRC for a full eight-bit character.

This is called the one-table, one-step look-up method. It is very fast but may be impractical because of the quantity of core required.

Two-32-halfword-table look-up method

A more practical subroutine for CRC character update relative to core storage requirements is the two-table method. In this method, we further partition the matrix X above into two matrices Y and Z . Thus we

TABLE I—Subroutine Using One-256-Halfword Look-up

Initial conditions for all subroutines:
 Register RA contains the old CRC, A_j
 Register RB2 contains the new character, C_{j+1} .
 Final conditions for all subroutines:
 Register RA contains the new CRC, A_{j+1} .

▽ CRC1		
EXCLUSIVE OR RB2, RA2	[1] RB[2:] ← RB[2:] ≠ RA[2;]	Form $W_{j+1}^{(L)}$
MOVE RB2, RA1	[2] RC[2:] ← RA[1;]	Form $W_{j+1}^{(H)}[0 I]$
SHIFT LEFT RB, 1	[3] RB ← ((15 ρ 1), 0) \wedge 1 ϕ (16 ρ RB)	Form address
LOAD RA, RB	[4] RA ← 2 8 ρ (16 ρ 2) \top SA[2 \perp RB]	Load $W_{j+1}^{(L)}X$
EXCLUSIVE OR RA, RC	[5] RA[2:] ← RA[2:] ≠ RC[2;]	Form A_{j+1}
ROTATE LEFT RB, 15	[6] RB ← 2 8 ρ (15 ϕ RB)	Reset address
BRANCH RETURN	▽	Return

write G^8 as

$$G^8 = \begin{bmatrix} 0 & | & I \\ \hline Y & / & Z \end{bmatrix}$$

Here, the Y and Z matrices are four by 16 binary matrices and $W_{j+1}^{(L)}$ is broken into two four-bit vectors $W_{j+1}^{(LL)}$ and $W_{j+1}^{(LH)}$. Thus, the new calculation becomes

$$A_{j+1} = W_{j+1}^{(LH)} \cdot Y \oplus W_{j+1}^{(LL)} \cdot Z \oplus W_{j+1}^{(H)} \cdot [0 | I].$$

Each of the products is a 16-bit row vector. The program now requires two look-up operations for the first two terms and a byte move for the last term. All three terms must then be EXCLUSIVE ORed together. The program is shown in Table II.

Binary summation method

Finally, it is possible to perform this whole operation without tables. This is done by performing the matrix multiplication by program rather than by table look-up. This requires a parity test as a condition on the branch instruction, however. This branching condition will be

called PTYRC, the even parity of register RC. Looking back to the defining equation

$$A_{j+1} = [C_{j+1} + A_j] \cdot G^8 = W_{j+1} \cdot G^8.$$

Let $D_k = [d_{0,k}, d_{1,k}, \dots, d_{15,k}]$ be the k th column of G^8 . Then the high-order position of the new remainder A_{j+1} is given by

$$a_{0,j+1} = \sum_{i=0}^{15} d_{i,1} \cdot w_{i,j+1}$$

which is operationally the same as ANDing the first column of the matrix G^8 with the working remainder W_{j+1} and finding the even parity of the result. This parity is the value of $a_{0,j+1}$. Similarly, we can find the remaining bits by ANDing W_{j+1} with each column D_{k+1} and find the even parity to determine $a_{k,j+1}$ $0 \leq k \leq 15$.

$$a_{k,j+1} = \sum_{i=0}^{15} d_{i,k+1} \cdot w_{i,j+1}$$

This operation can be carried out in a program as illustrated in Table III.

The program shown here requires more than 80 words

TABLE II—Subroutine Using Two-32-Halfword Look-up

▽ CRC2		
EXCLUSIVE OR RB2, RA2	[1] RB[2:] ← RB[2:] ≠ RA[2;]	Form $W_{j+1}^{(L)}$
MOVE RA2, RB2	[2] RA[2:] ← RB[2;]	Save $W_{j+1}^{(L)}$
AND RB2, H'FO'	[3] RB[2:] ← RB[2:] \wedge 1 1 1 1 0 0 0 0	Mask address
ROTATE LEFT RB2	[4] RB[2:] ← ϕ RB[2;]	Form address
LOAD RC, RB	[5] RC ← 2 8 ρ (16 ρ 2) \top SA[2+2 \perp 16 ρ RB]	Load $W_{j+1}^{(LH)}Y$
EXCLUSIVE OR RC2, RA1	[6] RC[2:] ← RC[2:] ≠ RA[1;]	$W_{j+1}^{(H)}[0 I] \oplus W_{j+1}^{(LH)}Y$
MOVE RB2, RA2	[7] RB[2:] ← RA[2;]	Get $W_{j+1}^{(L)}$
AND RB2, H'OF'	[8] RB[2:] ← RB[2:] \wedge 0 0 0 0 1 1 1 1	Form address
EXCLUSIVE OR RB2, H'10'	[9] RB[2:] ← RB[2:] \neq 0 0 0 1 0 0 0 0	Form address
ROTATE LEFT RB, 1	[10] RB[2:] ← 1ϕ RB[2;]	Form address
LOAD RA, RB	[11] RA ← 2 8 ρ (16 ρ 2) \top SA[2+2 \perp 16 ρ RB]	Load $W_{j+1}^{(LL)}Z$
EXCLUSIVE OR RA, RC	[12] RA ← RA \neq RC	Form A_{j+1}
BRANCH RETURN	▽	Return

TABLE III—Subroutine for Binary Summation Method

▽ CRC3		
EXCLUSIVE OR RB, RA	[1] RB←2 8 ρ(16ρRB) ≠(16ρRA)	Form W_{j+1}
LOAD RA, ZERO	[2] RA←2 8 ρ0	Set A_{j+1} to zero
LOAD RC, D1	[3] RC←SA[1;]	Load D_1
AND RC, RB	[4] RC←2 8 ρRC^(16ρRB)	Calculate D_1W_{j+1}
BRANCH [7], PTRC	[5] →(≠/(1, 16ρRC))/SECONDBIT	Branch if $a_{0,j+1}=0$
EXCLUSIVE OR RA, H'8000'	[6] RA[1;]←RA[1;]≠1 0 0 0 0 0 0	Set $a_{0,j+1}=1$
LOAD RC, D2	[7] SECONDBIT:RC←SA[2;]	Load D_2
AND RC, RB	[8] RC←2 8 ρRC^(16ρRB)	Calculate D_2W_{j+1}
BRANCH [11], PTRC	[9] →(≠/(1, 16ρRC))/THIRDBIT	Branch if $a_{1,j+1}=0$
EXCLUSIVE OR RA, H'4000'	[10] RA[1;]←RA[1;]≠0 1 0 0 0 0 0 0	Set $a_{1,j+1}=1$
And so on for the third through the 15th bits.		
LOAD RC, D16	[12] SIXTEENTHBIT:RC←SA[16;]	Load D_{16}
AND RC, RB	[13] RC←2 8 ρRC^(16ρRB)	Calculate $D_{16}W_{j+1}$
BRANCH [16], PTRC	[14] →(≠/(1, 16ρRC))/OUT	Branch if $a_{15,j+1}=0$
EXCLUSIVE OR RA, H'0001'	[15] RA[2;]←RA[2;]≠0 0 0 0 0 0 0 1	Set $a_{15,j+1}=1$
BRANCH RETURN	[16] OUT:→0	Return

of storage. However, a reduction in the storage requirement is possible by forming a loop to calculate the 16 binary sums. Further reduction is also possible when a specific polynomial is chosen and a combination of this and other schemes is used. For example, using $G(x) = x^{16} + x^{15} + x^2 + 1$, the number of instructions can be reduced to less than 20, making this method competitive with the other two given here. The key to this method is the branch instruction which tests the condition of the parity of the 16 bits in the accumulator. This is the last of the three matrix-oriented methods to be discussed and generally requires less core storage and more execution time than the previous two.

Other methods which partition the G^8 matrix in other ways are possible and may be better in specific cases.

SUMMARY

Using a matrix description of the operations required to generate the check bits in a cyclic redundancy error-detection scheme leads to new approaches to the software implementation problem. Certain variations are in use today and have proven to be superior to direct shift register simulation programs in most cases. With an apparent increase in programmable terminals and multiplexers, such approaches are likely to become even more important in the future.

REFERENCES

- 1 W L SCHILLER R L ABRAHAM R M FOX
A VAN DAM
A microprogrammed intelligent graphics terminal
IEEE Transactions on Computers Vol C20 No 7 1971

- 2 A W MAHOLIC H H SCHWARZELL
Integrated microprogrammed communications control
Computer Design November 1969
- 3 IBM 1130 synchronous communications adapter subroutine
SRL File 1130-30 Form C26-3706-4 IBM Corporation
White Plains New York
- 4 W W PETERSON
Error-correcting codes
The M.I.T. Press Cambridge Mass 1961
- 5 E R BERLEKAMP
Algebraic coding theory
McGraw-Hill Book Company New York 1968
- 6 I B OLDHAM R T CHIEN D T TANG
Error detection and correction in a photo-digital storage system
IBM Journal of Research and Development Vol 12
No 6 1968
- 7 R T CHIEN
Burst-correcting codes with high-speed decoding
IEEE Transactions on Information Theory Vol IT-15
No 1 January 1969
- 8 M Y HSIAO K Y SIH
Serial to parallel transformation of feedback shift register circuits
IEEE Transactions on Electronic Computers
Vol EC-13 pp 738-740 December 1964
- 9 A M PATEL
A multi-channel CRC register
AFIPS Conference Proceedings Vol 38 pp 11-14
Spring 1971
- 10 K E IVERSON
A Programming Language
Wiley New York 1962

APPENDIX

Here, we will show how a shift register is used to perform the functions required to generate or verify a code word (calculate the proper h bits of redundancy). Then it can be shown that the operation of a shift

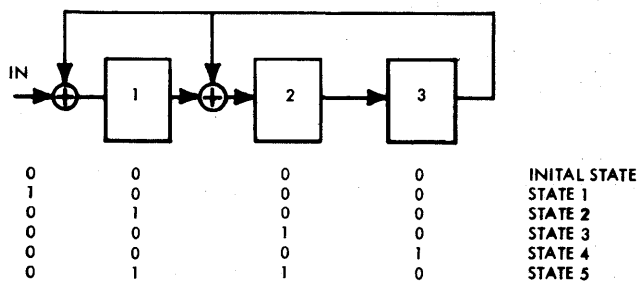


Figure A1—An elementary shift register

register on a bit-by-bit basis can be written in terms of matrix operations on vectors. Using this approach, it is possible to justify the several table look-up software schemes which are developed in the main text.

A feedback shift register is a device which stores bits in a serial string and is capable of shifting the string one bit at a time. There may be EXCLUSIVE OR and AND gates associated with the shift register which will operate when a shift takes place. The structure of a shift register is shown in Figure A1. The bit storage positions are indicated by a box (\square) and the EXCLUSIVE OR gates are indicated by the " \oplus ." If the storage positions are denoted as shown, we can illustrate the operation by assuming that bit positions 1, 2, and 3 contain zero and that a one bit is placed on the "IN" lead. A single shift of the register by a clock pulse (not shown) will cause the "IN" to be EXCLUSIVE Ored with the feedback from position 3 and placed in position 1. Thus position $1 = 1 \oplus 0 = 1$. Now, let us assume that "IN" is set to zero and then another clock pulse occurs. Position $3 \oplus \text{"IN"} = 0$ is placed in position 1. Position $1 \oplus \text{position 3} (1 \oplus 0 = 1)$ is placed in position 2.

A general shift register which performs division by

$$G(x) = 1 + g_1x + \dots + g_{h-1}x^{h-1} + x^h$$

is shown schematically in Figure A2. The AND gates are represented by the " \odot ." Although the output does represent the quotient, of major interest to us is the

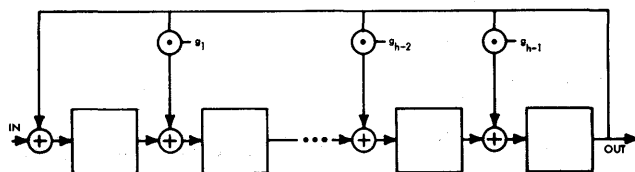


Figure A2—A general division shift register

contents of the shift register which is the h bit remainder

$$R(x) = r_0 + r_1x + \dots + r_{h-1}x^{h-1}$$

of the bits shifted in at any time. Thus, if we shift information bits

$$I(x) = i_0 + i_1x + \dots + i_{k-1}x^{k-1}$$

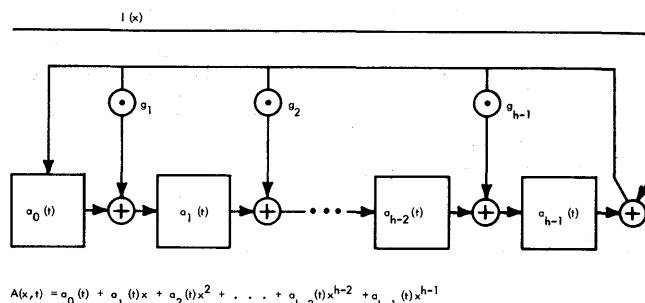
into the shift register, highest degree coefficient first, we will have the remainder of $I(x)$ after all k bits have been entered. However, we would prefer to have the remainder of $x^h I(x)$ rather than the remainder of $I(x)$ so that we may append the remainder bits directly to the information. One way to do this would be to shift the shift register h times after $I(x)$ has been entered. However, this represents wasted time since we can wire the shift register differently in order to cause it to "pre-multiply" by x^h . This shift register is shown in Figure A3, and the remainder at time t will be denoted by the polynomial $A(x, t)$. After shifting $I(x)$ into this circuit, the remainder $R(x)$ of $x^h I(x)$ divided by $G(x)$ will be contained without further shifts; that is, $A(x, k) = R(x)$. If $R(x)$ is appended to $x^h I(x)$, a code word will be formed $(R(x) + x^h I(x))$. At the receiver, exactly the same circuit or program may be used to determine whether the received block is a code word.

In order to further illustrate the operation of the shift register, it is possible to develop a set of functional relationships between the bits that have entered the shift register and the contents of the register. These are the circuit equations for the shift register.

Let the bits in the shift register (Figure A3) at time t be represented by

$$a_0(t), a_1(t), a_2(t), \dots, a_{h-1}(t)$$

where $a_0(t)$ is the leftmost bit in the shift register. We will also denote the bits which are shifted into the shift register as $b(t)$. That is, the contents of the shift register at time T include the effects of all $b(t)$ for $0 < t \leq T$. Since the bits come at discrete times, both



$$A(x, t) = a_0(t) + a_1(t)x + a_2(t)x^2 + \dots + a_{h-2}(t)x^{h-2} + a_{h-1}(t)x^{h-1}$$

Figure A3—A shift register for pre-multiplication by x^h and division by $G(x)$

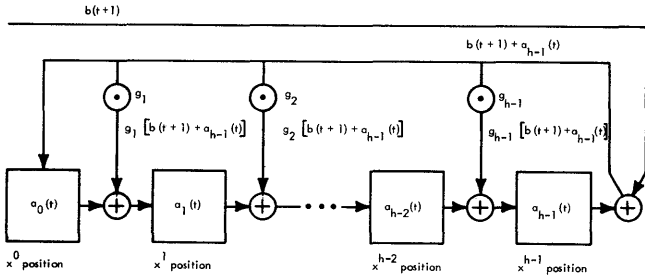


Figure A4—Development of circuit equations from the pre-multiply shift register

t and T are integers. Figure A4 may help the reader visualize this operation. From the figure, we can write the circuit equations directly.

$$\begin{aligned}
 a_0(t+1) &= b(t+1) \oplus a_{h-1}(t) \\
 a_1(t+1) &= a_0(t) \oplus g_1[b(t+1) \oplus a_{h-1}(t)] \\
 a_2(t+1) &= a_1(t) \oplus g_2[b(t+1) \oplus a_{h-1}(t)] \\
 &\vdots \\
 a_{h-2}(t+1) &= a_{h-3}(t) \oplus g_{h-2}[b(t+1) \oplus a_{h-1}(t)] \\
 a_{h-1}(t+1) &= a_{h-2}(t) \oplus g_{h-1}[b(t+1) \oplus a_{h-1}(t)]
 \end{aligned} \tag{A1}$$

Since we set the register to zero before beginning to calculate the remainder, we have the initial conditions

$$a_0(0) = a_1(0) = a_2(0) = \dots = a_{h-1}(0) = 0.$$

With these we can calculate any $a_i(T)$ given the $b(t)$ ($0 < t \leq T$) and the generator polynomial

$$G(x) = 1 + g_1x + g_2x^2 + \dots + g_{h-1}x^{h-1} + x^h.$$

These circuit equations will be used in the development of the matrix equations which are the subject of the main section.

In order to develop a matrix approach to the generation of a set of parity or check bits, we define a vector which consists of h binary components and represents the bits in the shift register at time t as defined above:

$$A(t) = [a_0(t), a_1(t), a_2(t), \dots, a_{h-2}(t), a_{h-1}(t)].$$

Next, we define G to be the companion matrix of the polynomial $G(x)$ as shown in the main text.

From the circuit equations (A1), it is apparent that

$$\begin{aligned}
 A(t+1) &= [a_0(t+1), a_1(t+1), a_2(t+1), \dots, a_{h-1}(t+1)] \\
 &= [0, a_0(t), a_1(t), \dots, a_{h-2}(t)] \\
 &\quad + [0, 0, \dots, 0, b(t+1) \oplus a_{h-1}(t)] \cdot G.
 \end{aligned}$$

Equation (A2) below follows immediately if one merely observes that

$$\begin{aligned}
 [a_0(t), a_1(t), \dots, a_{h-2}(t), 0] \cdot G \\
 = [0, a_0(t), a_1(t), \dots, a_{h-2}(t)].
 \end{aligned}$$

$$A(t+1) = \{A(t) + [0, 0, \dots, 0, b(t+1)]\} \cdot G. \tag{A2}$$

This is equation (1) of the main text.

Development of computer applications in emerging nations

by ALAN B. KAMMAN

Arthur D. Little, Inc.
Cambridge, Massachusetts

INTRODUCTION

The purpose of this paper is to explore a series of guidelines which will help identify attractive computer applications in emerging countries. We will examine areas of development from the standpoints of natural resources, labor intensive industries, public and private services. Then we shall explore levels of development in communications, education, high technology and the financial commitment of a nation. We shall examine the computer applications from a cost versus benefits viewpoint, and finally discuss a feasibility planning study.

A basic assumption throughout this paper, is that there is no such thing as a "model" or "average" developing country. Nor is there such a thing as an "average" computer application. Each emerging nation and each application must be explored and related as a separate case.

AREAS OF DEVELOPMENT

Natural resources

Developing countries can often be placed into one of two major categories; those that primarily depend on natural resources for their economy, and those which depend on labor intensive industries. From the standpoint of natural resources, the subcategories include applications such as agriculture, fuels, minerals, water and power. South American copper and coffee countries, West Indian fruit companies, Near East oil producing countries provide relevant examples.

Computers often enter first into natural resource countries with high technology industries (to be discussed in more detail later) such as in oil producing lands. Aruba, whose main economy is based on the Lago oil refinery, an affiliate of Standard Oil of New Jersey, shows such an application. The oil company purchased the island's first computer in 1961, and used

it for both accounting and engineering applications such as the critical path method, plant loading and blending, and standard accounting functions. The hardware served until late 1967, when it was replaced by a third generation machine. Antigua also saw its first computer installed at the West Indies Oil Company doing applications on profit/output relationships and linear programming.

The need for computers in agriculture is often overlooked by industrialists who are not familiar with the economic needs of countries depending on this natural resource. Greece, a predominantly agricultural country, provides an example. Almost half of the population derives a living from farming or farm-related activities. The farm population is about 4 million people distributed among some 1.1 million farm holdings. Farm sizes are very small, and often not enough to support a family adequately. About 60 percent of the farms range from one to ten acres in size, and most of these are not irrigated. The large-scale farm does not really exist in Greece.

Major farm management decisions in planning by the Government for the effective use of a country's agricultural resources depend on determining the optimum use of their scarce agricultural resources; primarily land, labor and capital. Traditionally, by this is meant planning the land use or cropping systems; planning the livestock enterprises compatible with that cropping system, and third, adjusting other resources in order to realize optimum returns for the total bundle of resources. Proper solutions to the resource allocation problems are vital to realizing optimum returns from the farming operation, the planned development of a country's agricultural sector, and the overall development of the country. The use of operations research techniques in analyzing the optimum combinations of the scarce agricultural resources for the individual farm, firm or government planner is both feasible, practical and valuable as a decision-making aid. The computer greatly facilitates this research.

In 1964, the computer presented to the Indian Agricultural Research Institute in New Delhi helped scientists to develop new seeds of wheat and sorghum, and a formula for the best conditions of sowing, fertilizing and irrigating.

These scientists had to go through thousands of "crossings" before hitting upon the correct genetic combination. The combination had to have a high yield, while being resistant to pests, diseases and climatic variations. This time-consuming process would have been practically impossible without a computer.¹

Other computer applications for the natural resource countries include the projection of fuel and mineral reserves, including geophysical exploration and development. The distribution of power and water by computer has already been undertaken by Russia, and is in need of implementation in such water-scarce countries as East Pakistan and parts of India.

Labor intensive industries

For countries with few natural resources to export, the key to their success is labor intensive industries. Examples include manufacturing companies and works programs.

Computers in labor intensive countries have generally followed or replaced conventional punch card equipment, with government taking the lead in their introduction. It is quite noticeable that the private sector has been slow to take the plunge, and most frequently it is those corporations with international connections that have installed the first computing equipment. Initial applications include inventory control, disbursement and revenue accounting applications. More creative programs in existence in developing countries include the pert-charting of major construction jobs, and the programming of construction logistics such as stress analysis, cut/fill balances and operations research. The most sophisticated step in these manufacturing operations is the use of small computers for numerical control applications. Here there is a danger of increasing unemployment unless the country itself is in a rising economy where displaced persons can find jobs.

In Romania, EDP was introduced in the early sixties, directly as a result of the fact that large-scale automation was introduced in production processes. Since that time, automation has increased by a factor of two, and a 450 percent expansion is envisioned during the 1971-1975 period.

Applications for this labor-intensive country include centralized control equipment for supervising the extraction and transportation of gas and oil, and the automation of hydroelectric stations. In Romania's

iron and steel industry, more than 90 percent of 1970 products were turned out by automated systems. The machine-building sector also has witnessed computerized control, with the truck works of Brasov and the Heavy Machine Works of Bucharest (UMGB) leading the way.

A priority control program exists for placing numerical control applications into this machine tool industry. Cement works, glass factories, weaving mills, footwear factories and the food industry (bakeries, breweries, sugar refineries, slaughterhouses) all provide relevant examples in Romania.²

In Finland, also, process control computers are expected to be in high demand from the rapidly expanding metal, engineering and chemical industries. EDP imports, estimated at \$9,000,000 in 1969, are expected to reach a level of \$21,600,000 annually by 1974.³

Public and private services

While countries fall easily into one of two categories (natural resources or labor intensive) for categorization of their export possibilities, a third major sector in each type of country can make use of computers to good advantage. Examples of the public and private services sectors include administrative government, military applications, transportation, communications, trade and commerce applications, financing and banking, libraries and education, health, social welfare and law enforcement, and finally, computer service bureaus.

Computers will choose people to fill 35,000 job vacancies in Ceylon, Colombo's public service this year. Furthermore, the hardest working computer in the country belongs to the Department of Census and Statistics. It processes data for the Registrar General, Police, Department of Education and the Customs Department. The Inland Revenue Department has undertaken a study to see how to utilize a computer in the most optimum manner when its new "pay-as-you-earn" tax reform goes into service.⁴

Needless to say, computer applications in these areas of priority development run a large gamut, including the standard disbursement and revenue accounting applications. One must remember, however, that in many developing countries in their early stages, the entire payroll is done in cash since a creditless society exists. Therefore, basic applications such as payroll, check servicing, etc., are not applicable. Banking applications, however, excluding check processing, still represent a formidable way of beginning. First, the statistical and numerical applications must be done accurately and in large volume. Furthermore, com-

munications between a main bank and its branches, can help in its own way to develop a communications system within the country.

The use of computers for a theoretical, rather than a mass production application is not common in developing countries but could prove to be immensely helpful. For example, many nations in their formative stages depend heavily on a series of strategies including alternatives based on a several-year plan. Pakistan has gone through four five-year plans and Algeria is in the midst of a four-year plan. The need for this is heightened by the fact that in a developing country, internal currency is usually worthless outside that nation's boundaries. Therefore, the nation must husband its foreign currency (such as dollars, pounds and francs) so that the exchange is spent on the most essential items for each nation.

It is common knowledge to people who have worked in developing countries that often the purchase of an automobile from outside is considered extremely wasteful and in some cases illegal. Penalties are placed on luxuries such as imported foods or alcohol, and in one country that I visited it was impossible to get a battery for my dictation unit because the batteries were considered to be luxury items.

To these countries, the strategies and alternatives to combine all the vast financial and human requirements of the entire nation require a synthesis, combination and analysis almost impossible by manual means. One of the greatest contributions to a developing country who is basing its entire economy on a plan, would be to set up and train the government officials in the use of simulation models directed at these planning applications.

To be practical, one must also recognize that several countries, far from a state of complete development, are using their government computers for defense applications. The use of EDP for military inventory and for war gaming is not limited to the developed nations of this world.

Of more practical use, is Nigeria's application where computers help them quickly recognize trends for epidemics that might be starting in geographical sectors of the nation. Furthermore, in 1969 approximately 14 computers existed in that country. Thirteen were being used in normal, industrial and commercial operations like payroll, billing and research connected with the oil industry. The fourteenth served the West African Examinations Council.

As its name implies, the Council's main function is to provide and administer examinations all over Nigeria. It is West African because the Lagos office is only a branch of the international organization whose headquarters are located in Accra, Ghana, and which

was set up jointly by the governments of Gambia, Ghana, Nigeria and Sierra Leone to conduct examinations in their countries. Besides conducting examinations, the Council was also a pioneer in the field of educational development.

To carry out its functions the Council makes use of two computers, one based in Accra and one in Lagos. The second computer would probably not be necessary were it not for the great distances and loss of time that would be involved in shuttling data from one country to the other. Source data comes chiefly in the form of candidates' entry forms. From these documents are produced lists which are required before an examination can be conducted, such as a packing list to enable officials to determine what quantity of materials and examination forms in each subject must be sent to the Center, a candidate list which can be used as an attendance sheet, individual timetables, and admission notices to enable candidates to know where they should report for the examination.

After the examination, source data comprise marked scripts and marked sheets from which marks for each candidate are punched. From these, mark distributions are made to determine the level of overall performance. The grades of each candidate and each subject are computed and the final test rate is determined by the machine. Eventually the results are listed and the certificates are printed by the computer from summary cards.

Computer growth in South Korea has risen from 20 to 30 in the past year. Back in 1967, the first two computers were imported by the Productivity Center and the Economic Planning Board. Furthermore, the Ministry of Science and Technology, set up in the same year, organized the National Computer Center to help facilitate usage. Currently, 32 percent of the computers are in operation in Government offices, 32 percent by the Universities, and 25 percent by special agencies. Industry and banks shared the remainder.

As of the end of 1970, Taiwan had installed 28 computers, of which 11 are used by Universities. The first was installed in 1964 at National Chlaotung University. An additional seven machines are used by the Government, including the Army, Navy and Air Force Logistics Commands.⁴

LEVELS OF DEVELOPMENT

Communications

Now that we have discussed priority areas of development, we should come down one step to talk about general levels of development within the nation under

study. Four major guidelines should be observed and the first of these is the field of communications. This includes the state of development of the telegraph and telephone system and the post office.

Any country, developed or undeveloped, will tell you quite quickly the state of their telephone or telegraph lines. As soon as the first computer system utilizing communications links is placed in service, the review will become much more critical. In general, the government or an airline becomes the primary group to experience problems. As discussed in the Nigeria case, two computers were necessary only because communications between two principal cities was virtually impossible within a reasonable length of time.

Not only, therefore, is instantaneous communication a problem for real-time systems, but the post office (or other means of carrying documents) often determines the application. It becomes almost useless to save time through using a computer in the central branch of a bank, if it takes three days to get documents to that center from one of the branches and another three days to return them.

In Algeria there is a major problem in communicating from the southern part of the country across the Sahara to the northern industrial cities. Many of the oil installations sit in the Sahara with very few links outward. Furthermore, the PTT has no immediate plans to extend major relief to the southern sector because of the problems involved and the lack of major usage. Here, a developing country finds itself in a chicken/egg relationship. Would the usage increase if the facilities were there? The answer in one case was a resounding yes!

The Telephone Department of the Government of Pakistan installed a direct dial cable between Lahore and Karachi to relieve the operator circuits between those two cities. They designed the size of the cable based on what they felt was necessary for traffic relief. They never anticipated that so many people just did not make calls because they found the service impossible. When the new link was opened, circuits were completely busy from four to six hours each day, and the public, if anything, became more frustrated with the addition of those new facilities because they were not able to get through on them.

Waiting time for telephone installations ranges from three to five years in some countries, and businesses are often required to buy their own switchboards and resell them to the Government Telephone Department at partial cost, then pay a monthly maintenance charge to keep them in service. Such conditions do not facilitate any type of computer usage where either source documents or output must travel great distances.

On the brighter side, reports from Taipei indicate

that a U.S. Air Force satellite program is being installed at the Linkou Air Station, where 18,000 punched cards record 6,000 supply items needed by Air Force personnel. A remote-batch computer at Linkou will use private line circuits to talk with the main computer at Ching Chuan Kang Air Base in Taichung. If an item is out of stock at Linkou, it will interrogate the larger supply base at Taichung. If the latter cannot supply it, the computer automatically orders it from the United States.⁴

Education

The need for education and a proper level of development during the initial introduction of computers within developing countries is essential. First, the vast majority of computer failures in developing countries occur because of a "love 'em and leave 'em" attitude. It would be embarrassing to tell you how much computer consulting work is done in developing nations because vendors have raced through the land selling systems, and then left the users with support ranging from inadequate to nonexistent. A program to provide computer hardware without provisions for training the necessary software and maintenance personnel creates more problems than it solves.

Furthermore, it almost appears necessary for a person who wants to keep up with the data processing profession to be able to read English, French or German. Since U.S. business executives primarily deal with heads of industry and top-level managers in foreign lands who have this capability, they neglect to realize that a vast majority of the technical workers or lower-level management employees cannot read with facility, technical literature in any of these three languages.

The Brazilian growth rate for EDP will probably be 20 percent from 1970 through 1974. (1970 base was approximately \$14,000,000.) Medium- and small-scale computers are in greatest demand. Computer room peripheral equipment is also forecast at the same high growth. These include printers, MICR equipment, memory systems and, outside the computer center, a wide variety of terminals.⁵

In Brazil, the Society of Users of Electronic Computers have predicted that 200 additional computers would be installed and an additional 1200 qualified computer programmers and systems analysts would be necessary within a year. The government has attempted to deal with the situation by providing Fortran lessons, and classes in general concepts of data processing at the Universities, without charge, to members of the Mathematics, Engineering, Social Science, and Science

Departments. Also, a post graduate course in computer science, leading to a masters degree is available from the federal university.

Singapore has a different method of education. The Singapore Computer Society, with over one hundred members, has been extremely active in promoting DP activities. The computer firms support this society activity in large measure by encouraging their own employees to lead discussions and classes in systems analysis and programming.

Of course, in any country with computer potential, the vendors give a wide array of courses. Although only 30 computers are installed in Taiwan, one vendor offers training in basic concepts, computer systems, programming, operations and special applications programming courses range up to three months in duration.⁴

The United States has recently tried some low-key training during an EDP mission to Taipei, Djakarta and Singapore. Anticipating that questions would come from businessmen who were only beginning to learn about EDP, the Commerce Department, acting through the U.S. Trade Center in Bangkok, arranged for local speakers to participate along with the mission members. The synergistic reaction among the two groups proved immensely successful.⁶

Conversely, an eastern nation with whom I have worked, has training facilities only through an international agency and has made no attempt whatsoever to implement computer training in the universities or technical trade schools. First, of course, schools must exist in reasonable quantity. For example, a recent census showed the population of another country at roughly 94 million. Education statistics indicated that approximately 3,800 students were enrolled in engineering courses at the universities, while approximately 200 additional students were attending technical or trade schools. In other words, only .004 percent of the population was involved in higher technical training.

There is no doubt that basic computer courses can be introduced early in a student's career to expose him to concepts and give him interest in the subject. Detailed courses could be set up on the same basis as the ITU/UN communications schools so successful in emerging nations. Once again, however, the chicken/egg relationship must be observed. If the training produces a large group of people who, upon graduation, have absolutely no possibility to use their talents because of the lack of hardware developments, the courses almost become senseless.

Finally, there has to be a technological flair on the part of the young people growing up, or no courses of this type will be popular. Surprisingly, a government official in one developing nation stated that students are

growing up with neither the desire nor the qualifications to use their hands. Although technical and trade schools exist in that country, enrollments are dropping, even as the population increases.

Existence of high technology industries

Often a small number of high technology industries within a nation can provide the nucleus around which computer development can grow. This is most obvious, although not limited to, oil producing countries. For example, many of the smaller airlines have justified reservation system computers on a combination cost/country-training basis. To speak to a previous point, in many of these cases it was necessary to advise the airlines to wait until the communications facilities within their own country could support a reservations system.

Banks often serve as the computer nucleus of a nation. Many Managing Directors of these institutions, however, will tell you the sad stories of training programmers, only to have them leave for higher paying positions as that nation's manufacturing group started to install machines.

Although not falling strictly within this category, computers depend on high technology industries for their daily operation. For example, the power requirements of computers are quite stringent. Voltage variations due to inadequate federal power service will cause malfunctions if they exceed allowable limits. It's clear that anything that stops the supply of power (a prevalent malady in developing countries) also will halt completely computer production.⁷

Financial commitment of a nation

Most important to the growth of computers is the financial climate under which they can be installed. Once again, the worthlessness of currency outside the nation's boundaries is critical. In one country, computer equipment was first priced internationally, then subject to a doubling factor as a penalty for using foreign exchange (since the vendor would not accept local currency) and finally, subject to a 100 percent tax on the original amount. Users who wanted hardware paid triple price.

On top of that, a vendor was maintaining a certain number of machine models in that country. When asked to supply a higher model in the series, the company requested the equivalent of several hundred thousand dollars extra to staff a special maintenance force and carry spare parts.

Computers need a complete stock of parts close by. Often they cannot be flown in from another country because the ensuing red tape caused by "purchasing" in foreign exchange rears its head. The charge by the previously named vendor was probably justified because the company knew that a week-to-month delay in getting the machine "up" would not be tolerated by the customer even though it was the customer's own governmental regulations which caused the bottleneck.

Conversely, that same vendor is well known for selling its obsolete models to emerging nations for their local currency. This policy has a great many advantages. First, the hardware has proven itself over the years and maintenance problems have been identified and categorized. Second, a great many standard software packages exist for such models and with adequate planning, an emerging nation can get much more than its money's worth by buying such a computer with available utility and application software. Third, of course, those nations can get equipment by spending their unrecognized currency, thus saving their desperately insufficient dollars, pounds or francs.

While Singapore has had a mixed series of reactions with computers, one factor strongly contributes to their growth in the country; it's a free port and no duty is charged on EDP devices. However, until recently a duty was levied on carbon paper according to the area of the paper, rather than by the sheet. Until this was changed, printer-paper was a first class luxury item, rather than a negligible cost supply as it is in the U.S.⁴

COSTS VERSUS BENEFITS

Items to consider

Such discussions lead to the next category; cost versus benefits. First, a number of United States industries lose money on computer applications and a developing country can't afford to do that. One reason for the loss is that these U.S. corporations spread their applications thinly. Any country can concentrate on one, two or perhaps up to 5 percent of standard applications and do an excellent job in terms of technical and economic measurements. The United States proved this when they used the first generation of computer equipment. The problem comes when countries try to expand too rapidly, or add too much at one time. Therefore, "limitation" is the initial secret to having benefits exceed the liabilities.

Next, it is important to recognize exactly why computers are introduced in various locations. If one wants to lose money he should recognize in advance that he is

going to do so. For example, the use of a computer as a status symbol in an emerging nation is quite common, although its justification is hidden under the guise of "competitive necessary." We have seen this particularly in the case of airlines, where the only thing they have to sell is service, and while the computer won't reduce costs in a country where wages are low and unemployment is high, the appearance to the public of a mechanized reservations systems is important to them. No doubt, status might be a very good reason in a very few cases for installing a computer. The key is to recognize it and admit it.

In early development stages, as discussed previously, usually no credit system exists. Payrolls are constantly paid in the cash of the land, and checks are virtually unknown. Therefore, the most basic system installed in the United States becomes useless for a number of years in this emerging "checkless society."

Inventory control is another basic United States package, which begins to become valuable in developing countries. Most particularly, the control of high unit-priced items will tend to save dispersion of foreign funds. Inventory control isn't necessary if the items are relatively inexpensive, and a large unemployed labor force exists. If, however, the items are high volume and expensive, such as drugs, inventory control could pay off in making sure that proper utilization limits the need for foreign currency to purchase additional amounts until they are really needed.

One of the largest engineering organizations in India, the Tata Engineering and Locomotive Company in Jamshedpur, stated that as early as 1969 it was able to save \$8,000,000 in its inventory control by using computerized programs. In addition, in a different type of inventory control computers helped the Indian Railways locate hundreds of "lost" freight cars and coaches.¹

Earlier we discussed the obvious benefits where a vendor will sell his older equipment in local currency, and probably accept that same local currency for maintenance. Along the same line, we do not underestimate the impact of the mini-computer market. For initial applications, no longer is the \$100,000-on-up computer necessary. For reasons of staffing and finances, very few mini-computer manufacturers have entered emerging nations. We feel that the loss is primarily theirs, and that intelligent marketing combined with support would yield them a much higher profit than that accruing to major manufacturers who maintain large facilities in these remote locations.

Conversely, the mini forces must not follow in the footsteps of the "love 'em and leave 'em" salesmen. That trend is now well recognized, and new companies entering the field will be questioned in great detail

concerning their method of supporting the software as well as the hardware, and means of training the national personnel. The problems will be greater because emerging nations are becoming smarter, but the profits for the vendor and savings for the customer exist in this low-priced computer field.

One of the great danger areas where costs can exceed benefits is, of course, where people are displaced in an economy which already has a high unemployment rate. One must be careful not to overstate the situation because the masses of unemployed in many nations might be incapable of performing even the clerical functions which are considered replaceable by a computer. Therefore, the "replaceable" labor force may be a small percentage of those masses.

In an emerging country, especially one that is now considering computers, it is quite possible that the types of jobs handled by the "replaceable" clerks are multiplying in other sectors. While it may not appear so at first glance, a detailed study may indeed show that for the clerical level and above, there is a rising economy and jobs can be found.

However, trade unions in India have complained of the use of computers, contending that with the vast manpower available, there is no justification for automation. The Government of India has been responsive to these arguments, and has adopted a policy of a "gradual switch to automation."

Conversely, India's computer growth (there are about 150 machines in the country in 1971) has opened a new line of jobs. IBM states they have trained nearly 125,000 Indian technicians in programming and other computer disciplines. These graduates have found jobs in India, Canada, Australia, the U.S. and other locations.¹

Furthermore, a salary inflation can happen which throws wages askew. This occurred in the United States with engineers, then with high technology experts and finally with computer programmers. The crafts developed so quickly that people to perform needed technical functions were scarce. Fairly soon the salaries necessary to attract such employees placed them in much higher salary brackets than their peers in the same company. Dissatisfaction was the minimum condition which resulted.

Conversely, the wage structures in many government institutions have been guided by long-standing financial instructions and general orders. These were drawn up before computers came on the scene, and before new skills and techniques connected with automatic data processing were developed. It isn't surprising that such a wage structure becomes unrealistic, since such regulations regard programmers and machine operators

as just another set of clerical staff. Therefore, where the choice exists, these people migrate to private industry. Case after case has occurred where government and banks lose programmers to the airlines, oil companies and private institutions.

One must determine the true social costs of computer personnel, ranging from the ones who are paid higher than normal to the unemployed. Most generally in an emerging nation it is important to get people working, and this objective is diametrically opposed to the United States commercial computer installation where its primary objective is to reduce high labor costs.

One cannot stress enough the advantages from a cost/benefit basis of standardized software applications. An emerging country rarely has enough personnel to maintain and run their system; no less to design it. In one case, we had a difficult time advising an airline to use either the Univac or IBM reservations system, since these were the only industry standards. U.S. airlines can testify to the horrors of developing such a complicated process with a vendor who has never done it before.

To give another example, so many common inventory control packages exist that it is sheer foolishness to invoke the Not Invented Here (NIH) factor and design a new one. The first concern of an implementation manager in a developing nation should be to collect all the available packages capable of being run on the computer for the application that he wishes to place. These should be stated when performing a feasibility planning study prior to either approving the application or ordering the hardware. As a matter of fact, in some cases the available packages might even dictate the hardware to be acquired.

Finally, we stress again that the use of computers for planning purposes is almost always considered a cost and rarely a benefit. This just simply isn't true if it can be utilized properly to develop strategies and alternatives for projects as important as national five-year plans. Operations research, simulation and modeling techniques have a definite purpose on a mechanized basis for a developing country. It would be well worth the investment to train the ministerial levels and their subordinates in the acceptance and use of such techniques.

Finally, the NIH factor often comes into play to prevent cooperation in developing EDP expertise. Perhaps because of the administrative functions involved there is very little coordination between users. In Nigeria, for example, when 14 computers existed (1969) there was no coordination or any centralized services in any way. Basic information was scanty and everyone thought they were developing their own

technique first. Hence redundancy in this labor-critical area occurred with all of its resulting waste.

The use of service bureaus gives an opportunity to many government departments and industries to trade valuable information and to "cut their teeth" on data processing techniques. It also provides an excellent training ground and a transition system while developing one's own in-house equipment. Sometimes the first industry to get a computer will make it available to others. ICL has been one of the leaders in convincing its customers to do this in emerging nations.

However, service bureaus on an independent basis could provide one of the best ways to start a country on the EDP path. Expertise in terms of both software and hardware would be centralized, and a systems design force could exist to be made available to all companies using the services. In general, the service bureau would operate on a batch basis, but it could be available to aid the telephone department in establishing initial installations of data processing lines.

Even developing countries must be allowed the use of a computer for pure pleasure. A computerized totalizer ("Tote Board") has been placed in operation at Djakarta, Indonesia's racetrack. It accepts data from up to 64 ticket issuing machines, and instantly calculates the odds. The "core" consists of three mini-computers and several multiplexors handling the input lines.⁸

The feasibility planning study

Since it is virtually impossible to develop one formula to relate areas of development, levels of development and costs versus benefits, the use of a feasibility planning study before hardware or software commitments are made is mandatory. At Arthur D. Little, Inc., a procedure has been developed by Thorpe E. Wright, and used quite successfully in emerging nations by this author. The process involves the formulation of an initial framework, then systematic expansion and recalibration to produce a finished document.

It is necessary to make sets of assumptions, see what results they yield, modify the original assumptions when appropriate, then see how this affects the results. Each of the six basic sections of the study may be developed independently, based on the sections which precede it. In a real sense, therefore, each section provides the foundation upon which the subsequent section is built, and therefore major additions or changes to any one section may affect any of the other sections.

The basic document, which is, at different stages of

its development, both a working document and a finished systems plan, comprises six basic parts.

Introduction

The primary importance of this section is that it establishes the need for the system. It should contain information concerning the history or background leading to and stating the need for the system. It may also include definitions of any terms used throughout the document.

Objectives

This section specifies the objectives to be achieved by the new system. It may also specify the manner or style of operation to be achieved or preserved by the new system. These objectives play a vital role in systems design since they provide the context within which various systems alternatives may be evaluated. Without them it is often not possible to resolve systems dilemmas.

Functional descriptions

This section contains statements of what the system is to do and the services to be provided to various classes of users. It also indicates in a general way the general response time requirements to be met, such as on-line response, daily processing cycle, etc. This section is wholly "what" oriented, with little or no consideration of how this is to be accomplished, and it is stated in non-technical terms.

Performance specifications

This section contains statements of the amount of work the proposed system must do. It includes such things as estimated numbers of key items to be processed, response time requirements for processing each of these key items, and the required reliability and operating performance for the system. Where the system has on-line terminals, it also includes estimates of the numbers of such terminals.

Design specifications

This section contains a proposed system of hardware and software capable of meeting the requirements stated in the previous three sections. The primary

purposes of this section are to assure that the system is technically feasible, and to design a realistic configuration to derive cost estimates for the system. Specifically, it contains rough estimates of overall system cost and time to complete the system. It is the most technically-oriented section of the six.

Feasibility analysis

This section contains statements of the four types of feasibility of a proposed system: technical, economic, acceptability to users, and legal acceptability. Technical feasibility is primarily comprised of statements concerned with whether the proposed system is workable and capable of meeting the specified performance requirements within the required time frame. It is also concerned with aspects of continuity of system performance where this is implied or stated in the performance specifications.

Economic feasibility is primarily comprised of various analyses and statements concerning the net savings (revenues or gross savings minus start-up and operating costs) and other tangible and intangible net benefits (advantages minus disadvantages) associated with the proposed system.

The third part is concerned with the acceptability of the proposed system services to users at various levels including the management level and the operator level. Where system users are outside the company, this might also involve marketing research studies. It also should involve a comparison of the final system design back to its objectives (Section 2) to assure that the objectives have been adequately satisfied.

This final category is primarily concerned with possible legal implications of providing the proposed system services. The government regulations of developing countries are often so rigid that system changes must be implemented to conform to them.

The feasibility planning process requires the close collaboration of two groups: user-management and the project study team. The user management is responsible for the content of certain parts of the study (specifically, Sections 2 and 3), while the project study team is responsible for the others, and may assist in the preparation of Sections 2 and 3.

The basic functions of the project study team are to make suggestions to the user management group, to do the staff work required to develop certain basic data about the system, and to determine the implications of various assumptions about what the system might do. The basic functions of the user management group is to assume the responsibility for the content of Sections

2 and 3 (Objectives and Functional Descriptions), and to make decisions concerning various system alternatives based on information presented to them by the project study team. Each group requires the other. The user management group is not normally capable of doing the required technical staff work whereas the project study team must carefully avoid making the required top management decisions or approving its own study results.

A key feature of the feasibility planning process is that it is only necessary that anyone who is not a data processing specialist, understand Sections 1, 2, 3, and 6 in order to understand fully what the system is and what its implications are. Since these key sections are written in non-technical language, it is easy for a person who is not trained in the EDP-related technologies to understand the system at any point in its development.

CONCLUSIONS

The purpose of this report has been to set out a series of guidelines on which to judge the most attractive computer applications in developing countries. It viewed the problem from three areas; priority development, levels of overall development and cost versus benefits. It discussed the primary division of countries into either natural resources or labor intensive categories, and added in a general sense the public and private services sector.

Under levels of development, it discussed four major categories: communications, education, high technology industries, and the financial commitment of the nation. Cost versus benefits were reviewed by setting forth a number of items to take into consideration, stressing that their applicability depended entirely on the nation and on the process under consideration. Finally, this paper gave a suggested method for performing a feasibility study before an emerging nation commits itself to hardware or other costs.

Key to the entire application of these ideas is the need to get away from generalities when discussing each problem. It has been the author's experience that no such thing as an average nation at an average state of development exists. It is extremely difficult to judge development on an overall basis since a country might be extremely well developed in one or two areas and backward in others.

Finally, and most emphatically, the paper stresses that computer applications should each be considered on their own merits; that standardized equipment and software must be used in the early stages and that maintenance of both the hardware and software must

be assured by the vendor before implementation begins. To this end, a feasibility study is essential both from a standpoint of justifying dollars and applications and from the standpoint of forcing its originators to set down in specific terms, the objectives, the approach, and ways of measuring the accomplishments.

REFERENCES

1 The New York Times
July 3 1970

2 Journal of Commerce
April 26 1971
3 Iron Age
October 1 1970
4 Far Eastern Economic Review
January 16 1971
5 Computerworld
November 4 1970
6 International Commerce
July 13 1970
7 Finance and Development
March 1970
8 Computer Digest
August 18 1970

Notions about installing and maintaining a population register in Brazil

by ANTONIO LUIZ DE MESQUITA

SERPRO/R. Eduardo Guinle 61
Rio, Brazil

INTRODUCTION

The problem of implementing and maintaining a centralized population register in a country as large as Brazil is a complex undertaking. This paper presents some facts and ideas underlying the work under way for the automation of the clerical and bureaucratic tasks of our government. A reliable Population Register system will undoubtedly be one of its cornerstones.

ENVIRONMENT

Status of data processing in government

Brazil is a Federative Republic of twenty two states comprising more than 4000 municipalities. At the federal level, most of the data processing is performed by SERPRO, a public company owned by the Treasury Department. It was founded in December 1964 by recommendation of the Administrative Reform Commission, in such a way as to encompass all of the data processing equipment and know-how then existing at that Department, including two 1401s and two UNIVAC 1004s, and employed approximately 40 professionals. SERPRO today has offices throughout Brazil and employs some 3000 people, dedicated to data processing. Its data processing equipment monthly bill is now of the order of 300,000 US dollars.

At the state level, only six out of twenty two state governments run their own data processing agencies, most of them organized as public companies, sometimes with the participation of private owners. Most of the state governments do not use data processing at all. The same is true for all but ten of the municipalities.

There is little use of data processing for defense. Computers are used by the military mostly for clerical purposes.

For federal and state government the two main applications developed are tax collection and payroll. The biggest success is in the area of income tax collection, controlled at the federal level.

From 1965 to 1970 the number of income tax payers grew more than twentyfold. In 1969, for the first time, the Treasury Department mailed back income tax refund checks, about 400,000 of them. This has increased to 900,000 in 1970. SERPRO has had prime responsibility for this breakthrough. There has also been a continuing effort toward the improvement of the quality of information by convincing public officials of its value.

No major advances have been introduced in the areas of data storage and data utilization. In the field of data processing, tape oriented and straightforward report generation techniques are still used.

The database concept, as well as some of the most recent tools of data utilization for management and planning, are now under study, on an experimental basis only.

Communications network development

In this area our federal government set up a special fund in 1967 to finance the improvement of the country's long distance communication network. A new company has been formed and it supervises the installation of about five million usable voice channel-kilometers in microwave linkages.

The program, costing some fifty billion dollars, is planned to become operational in mid-1972. The first benefits are however already here. A new breeze is blowing over our local telephone companies. New regulations have allowed them to be funded directly from the subscriber. Also the quality of long distance calls has improved and is exerting pressure on local services. The telephone system is thus becoming a reality in Brazil.

The mail system however has not kept pace. Only now are the first automatic letter dispatchers going into operation. The Post Office has been turned into a public company, and this will certainly help to improve future services. One third of the city of S. Paulo, the Brazilian huge industrial metropolis, still lacks the services of a postman.

Use of identification cards

Identification cards are sometimes not used in connection with Population Register. In Holland the Population Register does not inform the individual of his identification number.

In Brazil, despite the non-existence of a centralized register system, identification cards are used and citizens are required to display them often. Partly for this reason, there exists a reasonable number of public agencies legally empowered to deliver identification documents, though each has a well defined and distinctive prime objective in mind. Driver licenses, labor cards, income tax cards duplicate in many aspects the regular identification cards. Even the latter are not issued by a single agency.

If this number were unique and were carried on an identification card together with other identification information, it would assure local auditing of the number's use and a permanent feedback system through the individual's reporting to the government. The use of magnetic character printing for the identification number in the card would also avoid its misuse.

Our basic problem in this respect is therefore to unify all of the existing identification documents into a single, multipurpose standard identification card. The legal support for this matter has already been established and requires as of now just small changes.

The Population Register under study

A Population Register is, in my understanding, a system which basically allows substitution of a non-unique alphabetic person identifier (i.e., person names), by a non-ambiguous code number. The principal properties of this number are uniqueness, ease of use and universality. Uniqueness is its most important property and requires the sustaining services of sophisticated computer systems.

Computer hardware is so powerful nowadays it requires no longer code numbers to carry particular meaning, such as a digit for sex, two for birth date, etc. . . . This may have been a must for systems of some

years ago where the code had to play the role of addresses and retrieval keys. Today, sequential coding provides the necessary flexibility to management, and a measure of protection against privacy disclosures.

The ultimate goals of a Population Register are:

- (i) the simplification of administrative routines;
- (ii) the control of population on an individual basis and the full use of social legislation; and
- (iii) the reduction of the burden of the government over society.

To meet these objectives the Population Register has to assure:

- (i) uniqueness;
- (ii) universality;
- (iii) minimum delay between the actual event that generates data and its incorporation into the files; and
- (iv) suitable response time for a broad class of users.

Two types of information have to be maintained in the Population Register files:

- (i) the identification information;
- (ii) general purpose information, i.e., non-identification information interesting to a large number of users (such as address, education level, etc.).

There is a large spectrum of identification data about individuals. These data differ in their frequency of updating and change, easiness of collection and number of possible values. The selection of the identification data is a vital point in the design of the control system. For each selected identification set there is a measurable probability of identifying a unique person. This probability has to be as high as possible, provided:

- (i) the update and response times are not substantially degraded; and
- (ii) the system's cost and complexity are kept under reasonable limits.

The solution proposed for the Brazilian Population Register Control System incorporates the use of Master and Complementary Files. The latter serves the purpose of resolving indeterminacy questions which may occur when searching the Master File. Taking advantage of modern hardware (specifically direct access storage) the system is being conceived with the Master

File permanently on-line, and the Complementary File scheduled on-line.

The objective of the Master File with entries by name and identification number, is to maintain permanently a cross-reference between these two person identifiers. Sex, date and place of birth are the other identification data items maintained on-line. Compressing techniques for data compactation are necessary because the premium on secondary storage space is greater than in processing time. Names have to be normalized and sometimes shortened, in which case a name's complement is recorded in the Complementary File.

Database/Data Communication philosophy is employed all along the system's design, as we will enforce the use of file handlers and transaction oriented software. However, the system is planned to start operating in batch. Future transition to an integrated data processing system, requiring an on-line communication environment, will probably be attained without too large an effort.

The role of the Population Register Control System at that time will be to exercise control over population decentralized databases, both functionally and geographically. Among these databases we may count on having the Social Security Pension Plan, the Medic Care, the Income Tax, Labor Funds, and Popular Savings Bank, to mention just a few. The Central Control System will avoid redundant and contradictory data collection and storage, having it centrally controlled and maintained. Local databases will supply detailed information where needed. Centralized processing will consolidate data into higher levels of aggregation. The exchange of information among the local databases will also be assured and disciplined by the Central Control System.

SERPRO, being the largest data processing agency for the federal government, is the natural vehicle to pursue these plans and to turn them into reality in the next 4 or 5 years. Other government data processing facilities will make use of SERPRO's services via terminals. The system will become a nation-wide government information system.

IMPLEMENTATION

The usefulness of a Population Register is related to the frequency people need to report their identification. The quality of information, viz., its level of updating, accuracy, etc., depends on the pressures exerted over the system by its users. The larger the universe of users, the better we think the system will work and in

general, the higher the quality of the information it will provide.

To implement the system, a new identification has to be provided for every citizen. This must be done as much as possible in accordance with the existing body of laws and regulations.

A practical way of issuing to a significant amount of citizens their new identification in a relatively short period of time is to make use of a simple sequential coding system. A census like campaign can achieve this goal. Pre-numbered identification forms will be distributed to the population thus tying the data collected about a person with the code number which has been assigned to him. During this phase it is not recommended to centralize geographically the assignment of identification numbers to persons, for this would slow down the impetus of the campaign.

Checks for code duplicates and for data validation would have to be carried out at file creation time. Also, provisions have to be taken to convert files containing the existing and varied identification information. Correspondence files will have to be established and maintained between the new and the old identification system throughout the duration of this phase.

At its end we will switch to a centralized code assignment operation. From this moment all data validation will have to be performed before a person is admitted to the Register. At this time the system may be fully operational, although expansions and adjustments will have to be expected.

Databases are fragile. Checkpoint and recovery procedures must be carefully thought out. Tape copies of the disk files have to be produced periodically. This is better justified if it takes place when exhaustive file searches become necessary to satisfy new requests.

There must also exist a single responsible institution for reporting updated information about each data item in the Population Register files. This is a key point in the updating process, where, once more, turn around time has to be very short.

CONCLUSION

Emerging nations must take advantage of their late start in many technological areas. In this regard, data processing in Brazil has to follow the steps we took in developing our long distance communication network. We started late, from scratch, and are making use of the most recent technology.

This must happen too in the field of data processing, mostly inside the government. In data processing the principal problem which must be solved is that of

manning the new technology in order to reconcile the requirements for software and hardware. This was not necessary in communications and in this respect implementation of the two technologies differ. One approach to solving the data processing problem is through local computer vendors. They must begin to

rely on local talents not only for marketing and manufacturing, but also for product development. This change in the rules of investment policy followed today will guarantee the catalytic element that will bring forth a locally developed technological society in this branch of activity.

The neurotron monitor system*

by RICHARD A. ASCHENBRENNER, LAWRENCE AMIOT and N. K. NATARAJAN

*Argonne National Laboratory
Argonne, Illinois*

INTRODUCTION

The subject of performance monitoring and measurement has grown from infancy to childhood, and with this growth came substantial performance improvements even with superficial monitoring analysis. The recent increased interest in applying measurement techniques by manufacturers and users of large systems stems mainly from the high cost of development, purchase, and use of such systems. This cost obligates each to obtain quantitative information on the dynamic behavior of proposed or purchased equipment and software. This quantitative information is necessary when a determination is to be made of the difference between potential and actual performance of hardware and software.^{1,2,3}

In addition, the particular areas of interest at Argonne National Laboratory which have benefited from the development of hardware and software monitoring techniques are: (1) configuration analysis and optimization; (2) "large" program profile analysis; (3) simulation analysis; and (4) computer architecture studies. Each area requires a parametric description of the system for solution. Most important, and even more elementary, is selection and quantification of the independent parameters or variables, rather than just of the measures which indicate performance for a particular situation. The methods used to evaluate and predict system performance must provide insight into how complex systems function; they must provide insight into the most important parameters and measures of the system; and they must also provide quantitative information on the sensitivity of these measures. System software monitors, program analyzers, and hardware monitors have each made their contributions in performance evaluation and prediction.

HARDWARE MONITORING

Hardware monitoring offers the ability to obtain information on system performance by directly attaching probes on a host system. Microevent analysis which would take inordinate time by means of timer trace simulation or gross statistics gathering can best be obtained by hardware monitors where event measurements can easily be selected and varied to minimize voluminous amounts of data recording.

Hardware monitoring has the obvious advantage of measuring systems which have no easily implemented means of software monitoring, or where the introduction of such artifact would cause system degradation. This is true especially in the computer-automated experiments and real-time or communications-oriented systems. Similarly, it is advantageous when the software artifact introduced affects the measurement statistics.

Previous studies using available hardware monitors demonstrated the need to obtain information directly at a more primitive level, greater in quantity, at higher bandwidths, and with more convenient and accessible output facilities. To this end, a hardware monitor project was initiated at Argonne National Laboratory to achieve a more creditable means of system measurement and evaluation.

This monitor has demonstrated its ability to interact with the monitoring process and to provide analysis and display concurrent with measurements. This monitor development has also aided in solving the problem of information loss due to sampling while reducing the data collection rates and raw data storage and processing requirements.

"Neurotron" monitor

The design of the Argonne "Neurotron" monitor overcomes the previously stated deficiencies in many hardware monitors, and in addition provides inter-

* Work performed under the auspices of the U.S. Atomic Energy Commission.

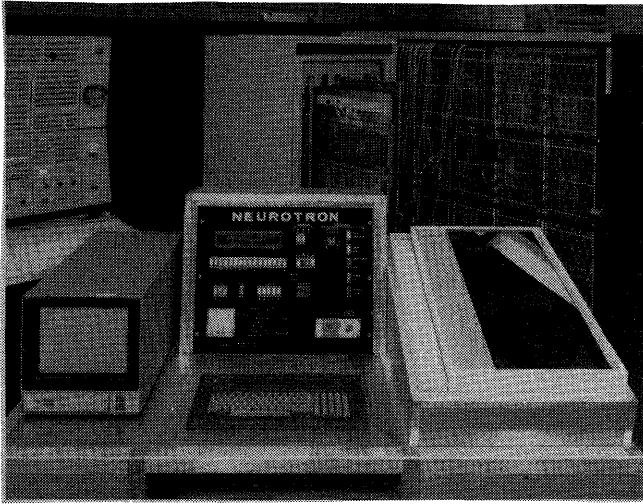


Figure 1—Photograph of the 'Neurotron' monitor

action by operator or program with the data accumulation, analysis, and display. This interactive (rather than passive) monitor is based around a minicomputer, storage display, tape unit, and specialized computer-controlled logic and data accumulation hardware. This interactive ability has also been the basis of design for the future communications between the hardware and software monitor processes.

The goals of the monitor development were as follows:

1. Capability of high bandwidth in logic and data accumulation facilities.
2. Program-controlled logic for selecting or filtering events based on current experiment or recently collected data.
3. Capability of obtaining data at subinstruction or instruction level as well as gross operating statistics.
4. Capability of response to events or interrupts of interest within a reasonable interval or to "automatically" select monitoring periods during events of interest.
5. Capability of recording and analyzing events or sequences with short (millisecc) perturbations as occurs in many "real-time" systems as well as presenting statistics on long-term variations.
6. Graphic output for providing messages and snapshots of the monitoring process to operators or experimenters.
7. Capability of obtaining information felt necessary for examining the highest performance processor locally available (360/MOD 75).
8. Inexpensive and portable as possible in order

that remotely located computer systems could utilize the equipment.

9. Relative ease in adapting hardware to new experiments or expanding the equipment as monitoring experience evolves.
10. Form a basis for a combination hardware-software monitoring process when further understanding of this process is available.

Figure 1 is a photograph of the equipment, and Figure 2 is a functional description.

The mini-CPU is used as the monitor control element. The computer coordinates the operation of the I/O, logic selection, algorithm selection for the Random Access Memory and arithmetic unit, programmed logic, counters and sequencers. This coordination is performed under a multiprogramming-priority system. Display programs, data acquisition programs, and analysis programs individually have priorities attached in addition to the priorities normally associated with I/O and probe interrupts.

Monitor elements

The basis for monitor data acquisition is the programmed selection and control of the elements in the monitor. In addition to the computer program selection of elements and paths, a 36x24 patchboard programmer is included to aid in the I/O selection.

Control elements

Programmable logic and registers which can be set and read by either the CPU or the external environment, form the main communication link. These

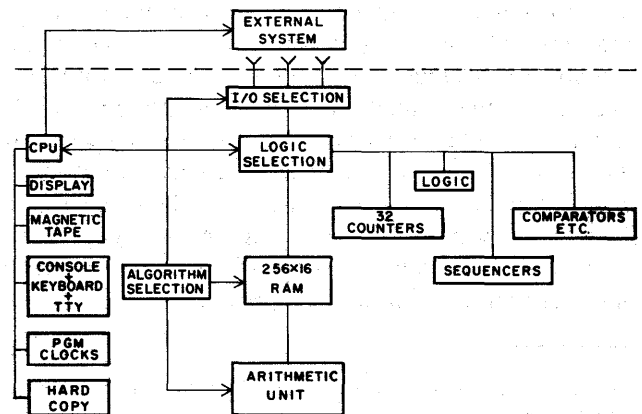


Figure 2—Functional description of the 'Neurotron' system

registers control the selection of input probes, logic selection, start/stop control, sequence configuration, transmission paths, and other control functions depending on the particular experiment in progress.

Logical elements

More typical of conventional monitors are combinatorial logic elements, decoders, comparators, and pulse generators. These devices are used in performing logical processing on the input signals. Thirty-two 40 MHz counters are available for event counting or timing. Each counter or group of counters can be selected by program for start/stop, read, or read and clear. In this manner, sampling intervals are completely at program discretion.

Sequencers

Sequencers are logical devices used in the determination of event occurrences relative to previous or subsequent events. Events may be addresses, instructions, device movements, encoded signals, etc. Each sequencer is designed to accept pulses representing an event and to track subsequent events. If a break in the defined series of events occurs, an output is enabled which can control other sequences, logical changes, or be used for counting or timing. Each physical device can be used for the sequencing of three events; however, sequencers can be chained to much longer lengths. Sequence detection experiments typically use from 2 to 16 events.

Random access memory (RAM)

A key element in the acquisition of data is a Random Access Memory and associated arithmetic unit. The RAM consists of 60 nanosecond access monolithic chips organized in a basic configuration of 256 words by 16 bits. Depending on the selection of the experiment, however, it can be used in configurations of 512×8, 256×16, or 128×32 bits. The CPU and external system can access this memory in several modes; e.g., write, read only, increment, and read/clear. In addition, control over external access is maintained by the CPU.

Data processing

Data processing is performed by combining and controlling the physical elements in the "Neurotron" monitor, accessing the contents of these elements by

```

CPU UTILIZATION          60%
CPU TIME, I/O OVERLAP   40%
TOTAL TIME, I/O ONLY    40%
SUPERVISOR STATE        50%
SUPERVISOR ACTIVE       25%
WAIT PENDING, 2301      30%
2301 ACTIVE (I/O TOTAL) 20%
CTC WAIT                 8%
MOD 50 USE OF SHARED FILE 85%
50 MUX USE BY REMOTE BATCH 8%
50 MUX USE 2821-V       85%

```

Figure 3—Text display of an activity interval

the CPU, and analyzing, recording, and displaying the effects of the monitored events.

Text displays

A typical output display is shown in Figure 3. The interval for sampling and the information displayed is selected by the operator or experimenter, constrained, obviously, by the monitored entities. Information can be presented in bar graph form if desired rather than in the text form illustrated.

Similar display and recording can be accomplished by encoding of events such as interrupts, device accesses, and channel use, limited generally only by the ingenuity of the experimenter.

Instruction analysis

In gathering statistics on instruction distributions correlated with I/O activity, time, program keys, or other events, the RAM is used as a 256×16 bit accumulator. That is, the instruction format (up to 8 bits) is used as the address field in accessing the memory. On each instruction execution (or instruction issuance depending on the host system architecture or statistic of interest) the RAM is accessed at the location specified by the address, updated by a count of 1, and restored to memory. The unit was designed to perform the update in less than 200 ns (sufficient for current equipment at the Laboratory). The sampling interval can be controlled by a CPU data collection program driven by a programmable clock, or can be determined by externally triggered or internally calculated events. By sampling interval is meant the time during which every instruction is monitored, counted, and totals are read into the CPU memory. Normally, at the end of each interval the data collection program can read or read/clear all locations of interest in the RAM as well as counters, communication registers, and other devices in preparation for the next interval while event counting

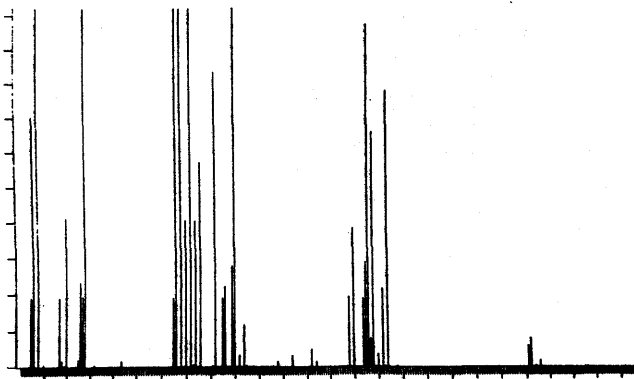


Figure 4—Distribution of instructions during sample interval

continues. The program may start and stop the collection of data during this interval, depending on the allowable skew between the reading of all data and continued accumulation (the degree of correlation).

Multiple samples can be retrieved, accumulated, recorded, and displayed. An example of an instruction distribution display for a sampled interval is shown in

TABLE I—Instruction Type Distribution During Successive Intervals

TYPE	INTERVAL I	INTERVAL II	INTERVAL III
SPECIAL, DECIMAL, EDIT	4.13 %	2.82 %	2.26 %
CONTROL, IO	.37 %	.38 %	.45 %
LOAD-STORE INTEGER	37.13 %	41.13 %	45.36 %
INTEGER ARITH.	10.24 %	9.72 %	7.76 %
LOAD-STORE FLOAT. PT.	2.42 %	1.94 %	1.45 %
FLOAT. ARITH.	.83 %	.82 %	.09 %
BRANCH	27.86 %	26.04 %	24.69 %
LOGICAL, TEST, COMPARE	17.02 %	17.15 %	17.94 %

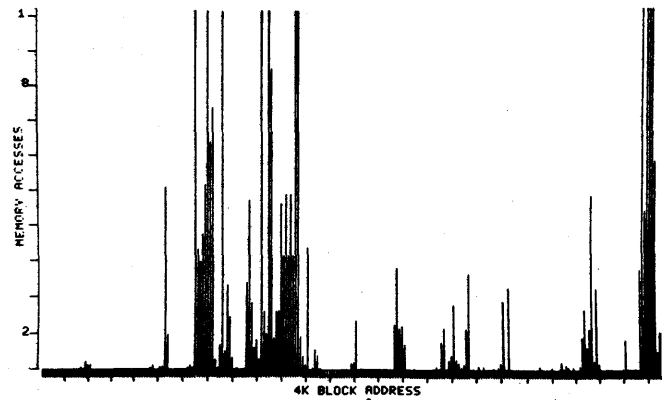


Figure 5—Memory activity display—interval 1

Figure 4. Selected portions or a condensation into major categories can be displayed if desired. A condensation for a time period including the displayed sample interval is shown in Table I.

Memory utilization

A similar approach is taken in monitoring address streams. Since the current size of the RAM is 256 words, only 8 bits of an address stream are utilized. In a one-million byte system, the host memory is partitioned therefore into 4K byte blocks. Any memory access of the host system increments the corresponding location in the RAM. Concurrently this absolute memory activity during a sampling interval can be retrieved, recorded, or displayed. Examples of a memory activity display for two consecutive sampled intervals are shown in Figures 5 and 6. Each division on the horizontal axis represents a 4K block of core, and the vertical displacement (full scale=10⁶ accesses)

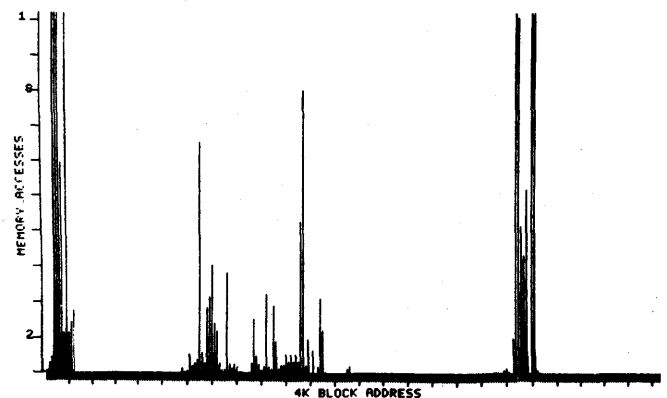


Figure 6—Memory activity display—interval 2

represents the absolute number of accesses made to that block in the sampled interval. The build-up and decay of utilization is quite obvious from these successive displays. Since the displays are under program control, scale changes and interval selections are available to the operator while absolute counts and correlated information from counters or sequencers are also displayed and recorded.

Memory accessing

In analyzing memory accessing in various systems, it may be more important to know the read/write characteristics and the relative magnitude of accesses than the absolute memory utilization. In this type of experiment the RAM is used as a 512x8 bit memory with even locations used for 'read' counting and odd locations used for 'write' counting. Thus, 255 counts may be accumulated in any sampling interval with counting inhibited after 255 is reached. While these counts are being retrieved and recorded, an operator display is generated, as shown in Figure 7, in which each block corresponds to a 4K byte segment of a one-million byte memory. This display indicates those regions active for read, write, or both, during the preceding interval and also indicates those regions with no memory activity. These latter regions may be allocated but are not active. Similar displays can illustrate channel and/or CPU access in each region.

Activity graphs

While statistics on memory or instructions are being collected, other information can be recorded in high-speed counters during the corresponding interval.

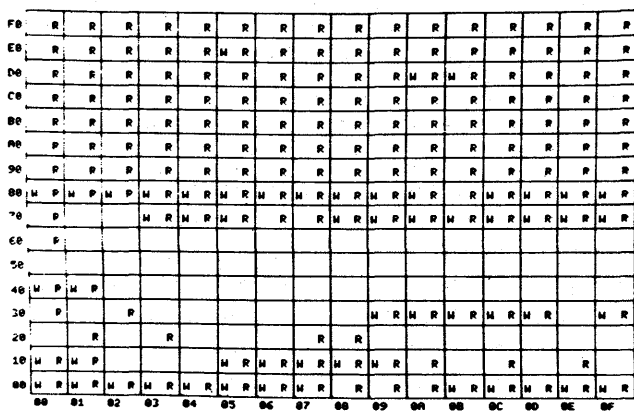


Figure 7—Memory access display—read/write activity

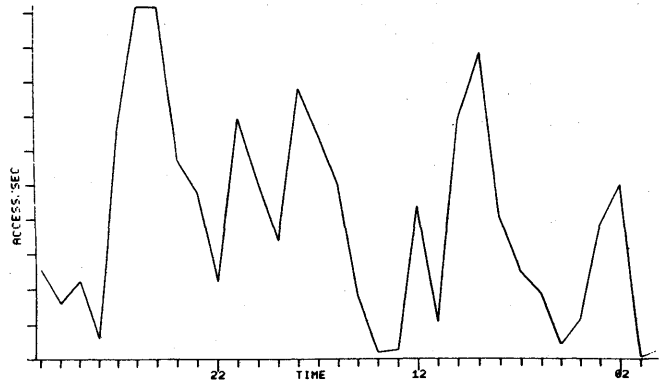


Figure 8—Device activity history

Updated activity graphs can be generated by the data acquisition programs indicating the most recent history of the device or event of interest. Displays similar to Figure 8 (indicating device activity in this case, for the last 32 seconds) can be most interesting and useful to operators and experimenters alike, especially during periods of high activity.

Buffer analysis

Since the RAM can be loaded from either the CPU or an external system, the memory and associated arithmetic unit may be used as a large quantity of comparators.

An example of this type of operation is in the analysis of buffer type memories and their appropriate algorithms.^{4,5} It is anticipated that hierarchy memories will be usefully implemented in a variety of design situations and it is necessary to determine their effects on a range of applications and environments. One useful technique currently implemented is called "congruence mapping." This technique has advantages both in ease of implementation and access time relative to mapping techniques necessitating a full associative search. Buffer configurations are N x M blocks of B bytes capacity each. The "Neurotron" allows the simulation of buffers up to 4 x 128 blocks. The selection of the address bits monitored determines the block capacity. The Random Access Memory is organized into a 128 x 32 bit memory, each word divided into four fields of up to 8 bits each. Obviously, any submultiple of each dimension can be used also. The number of successes (buffer hits), the class level of the match (or replacement) and other information is available for recording for the currently implemented algorithms of least recently used (LRU), first in-first out (FIFO), and random replacement. Again, the sampling interval can

$N \times M \times B$	REPLACEMENT ALGORITHM	TYPICAL SUCCESS RATIO
4 × 128 × 64	LRU	97% - 99%
2 × 128 × 64	LRU	94% - 98%
4 × 128 × 32	LRU	97% - 99%
2 × 128 × 32	LRU	91% - 98%
4 × 128 × 64	FIFO	95% - 99%
2 × 128 × 64	FIFO	94% - 97%
4 × 128 × 32	FIFO	92% - 96%
2 × 128 × 32	FIFO	89% - 95%

TABLE II—Typical Results Obtained by Address Stream Monitoring, with 'Neurotron' Used as a Simulated Buffer Memory

be determined by program, clock, interrupts, events, etc.

By monitoring the various address-generating mechanisms in a system without buffer capabilities, the effects of buffer size and replacement algorithm on such equipment can be determined. Table II indicates results of a few randomly sampled intervals on a S/360/MOD 75 with 1M byte of fast core. Similar experiments have been performed on computer control equipment, communications concentrators, and time-sharing systems. Variations of this experiment are used in analyzing use of distributed and read-only memory.

Data recording and display

Once data has been retrieved from counters, RAM or communication registers, information buffers may be updated, recorded, or displays generated. By use of rotating buffers, double buffering and other techniques, statistics with a resolution of a few milliseconds can be recorded, or significant filtering and compression of data can be performed before display or recording.

The interactive display has provided a means of pre-acquisition probe adjustment, judging the reasonableness of the on-line data acquisition and reduction programs, and snapshots of the performance statistics during data collection. It also provides information to inquiries by means of messages, histograms, bar charts, time plots, etc. This interactive capability has effected

a time savings not only during monitoring, but is useful also in observations during post monitoring analysis which may be performed in the monitor itself.

As anyone familiar with hardware monitoring techniques can verify, the probing of a large number of unfamiliar systems can be difficult. A display of the information currently being processed has proved useful in determining that probes have been properly placed and are in working order. Similarly, since the data acquisition programs are usually time or interrupt dependent, the display can provide a means of judging the necessary sampling intervals or buffering techniques for the proper display and recording of data before final monitor results are obtained.

Snapshot displays of the statistics are useful to experimenters and operators interested in more immediate information before postmonitoring analysis. This information usually relates to device utilization, interrupt activity and associated core utilization, and channel activity.

This immediate data reduction can also potentially provide feedback to software monitors executing in the host system. Currently being designed is a channel interface to IBM 360 equipment. It is felt that this interface between our existing monitors will provide a means for a more optimum collection of information for both system and user programs. Our experience thus far, however, has demonstrated the usefulness of interactive hardware monitors in several environments in which no software monitors are available, or in which the event bandwidth is outside the capability of those monitors.

Software development

An on-line operating system for the "Neurotron" was developed⁶ to service the diverse applications anticipated. This operating system establishes an environment for allowing a versatile priority structure to be defined by the user programs. Although physical interrupts and devices within the system are assigned separate priorities, each program or module may have separately assigned execution priorities. This allows users to have dynamically varying priorities for various modules based on current data rates, event occurrences, program type, etc. The actual scheduling of programs and interrupt connect/disconnect can occur by means of keyboard input, interrupt occurrence, or from another routine. A rigid modular structure allowed the operating system to be highly interruptable and greatly decreased the development time. With the operating system provided a set of routines to aid users in the develop-

ment of their applications. Programs for display, dump, trace, interrupt connect, breakpoint, etc., are available for debug and on-line use.

CONCLUSION

The development of the Neurotron has provided the engineers and system programmers at Argonne with a convenient means of monitoring the operation of a variety of computing facilities. The monitor organization and acquisition hardware have allowed the recording of data whose collection heretofore was prohibitive, expensive, or time-consuming. The interactive and display capabilities of the system have provided the user with the necessary facility for immediate interrogation and presentation of data. It is estimated that similar monitors could be made commercially available for \$35,000 to \$65,000 depending on software provided, logical features, etc.

Data of the type previously shown is continually available to the user to provide the necessary operating picture of the system. The monitoring work that has been accomplished to date suggests the usefulness of a

real-time interaction between system monitoring (hardware and software) and system programs such that dynamic system adjustment is both possible and useful.

REFERENCES

- 1 G ESTRIN et al
SNUPER computer
AFIPS Spring Joint Computer Conference 1967
- 2 P CALINGAERT
System performance evaluation: survey and appraisal
Comm ACM Vol 10 No 1 January 1967
- 3 D HOPKINS G ESTRIN
An interfering instrumentation computer
UCLA 10P14 57
- 4 L A BELADY
A study of replacement algorithms for a virtual storage computer
IBM Systems Journal Vol 5 No 2 1966
- 5 R L MATTSSEN et al
Evaluation techniques for storage hierarchies
IBM Systems Journal Vol 9 No 2 1970
- 6 L AMIOT R ASCHENBRENNER
The 'Neurotron' operating system
Argonne National Laboratory Applied Mathematics
Division Technical Memorandum No 223 unpublished

A simple thruput and response model of EXEC 8 under swapping saturation

by J. C. STRAUSS

Washington University
Saint Louis, Missouri

INTRODUCTION

EXEC 8 is the multiprogramming, time sharing operating system for the Univac 1100 computer systems. EXEC 8 attempts to provide satisfactory concurrent batch, demand (interactive), and real time processing through complicated priority scheduling schemes for both real memory and CPU time allocation. Basically, the scheduling schemes allow real time service to have whatever resources it requires and demand and batch service requests share the remainder. The sharing algorithm is quite complicated; in essence, however, it dynamically limits the time average impact of demand service on the system performance to an installation set limit function of the number of active demand users. Within the demand and batch type categories, time and core are allocated by exponential scheduling algorithms biased to favor small jobs, but constrained to service all jobs eventually. In addition, EXEC 8 provides all the I/O control, file handling, diagnostic error testing, user support systems, etc., normally associated with third generation operating systems.

This paper presents a simple deterministic steady-state model developed to help understand the gross scheduling and resource allocation problems in the operation and (particularly) the performance tuning of EXEC 8. The model is concerned solely with the long-term balance of demand and batch services; as such, it does not concern itself with real time services and need not concern itself with the standard operating system user services.

This is but one of a number of attempts¹⁻⁷ to model significant behavioral aspects of very complex computing systems by simple models. Most of the referenced papers present justification for the philosophy. To avoid repetition here, suffice it to say that the problem is basically no different than any complex system modeling problem; i.e., groupings of interesting and

significant behavioral aspects are isolated to preserve, in some sense, a homomorphic mapping between the original system and the reduced model. The verification and interpretation problem is also similar to other modeling and simulation situations; i.e., the model is verified for measured behavior and employed to predict unmeasured and/or unmeasurable behavior. Here, too, the significant problem is to limit the aspirations of the study and not attempt to employ the model in situations that do not preserve the original homomorphic mapping.

This work on the performance model presented here developed out of a larger system performance evaluation and timing study concerning the 1108 operated by SINTEF (a non-profit engineering research foundation) for the Technical University of Norway (NTH), Trondheim, Norway. This study was prompted by the circumstance of upgrading from a very satisfactory Univac 1107 to an 1108 and experiencing an increase in the installation cost/performance ratio. This was subsequently explained by a number of factors such as lower discount percentage on the 1108, minimum EXEC 8 configuration, workload tuned to the 1107 EXEC 2, etc. However, this post facto analysis did little to soften the blow. Also, in trying to analyze the behavior of EXEC 8 with a view to tuning the system control parameters for "optimum" performance with the local workload and configuration, it was determined that Univac (at least in Europe) did not understand EXEC 8 very well. Thus before initiating more ambitious measurement, analysis, and simulation projects aimed at performance tuning, it was necessary to obtain simple conceptual and analytic models of significant behavioral aspects of the system.

The model developed is based on the not unreasonable assumption that under heavy pressure for demand service the single channel swapping device of the NTH 1108 configuration will saturate and thereby become the limiting resource to system performance.

Most other simple models are based on limiting resource assumptions of one sort or another. For example, in Reference 7 Kimbleton and Moore develop a simple model of IBM 360/67 performance based on the assumption that the CPU is the limiting resource. While limiting resource models are certainly conceptually and often analytically simple, their usefulness is very much dependent on the validity of the original limiting resource assumption. The validity of the swapping saturation assumption underlying the current model is investigated here in concept, by measurement, and finally directly in terms of the model parameters.

The extent to which the model is a success has to be measured against its initial goals; i.e.: (1) to follow and predict steady-state EXEC 8 performance as a function of configuration and workload, and (2) to serve as a focus for detailed study of EXEC 8 design, construction, and behavior. Both these points are discussed in the sequel in light of presented results.

This paper is organized as follows: the next section describes the manner in which the load is characterized and presents those features of the EXEC 8 core and CPU time scheduling algorithms that significantly affect the average behavior of the system. The BASIC MODEL section develops the basic model and the VERIFICATION section attempts to verify this model against behavior observed at NTH. The AUGMENTED MODEL section analyzes the shortcomings of the basic model and proposes and verifies an augmented model designed to correct problems due to limited core space. The final section analyzes the region of significance of the swapping saturation assumption underlying both basic and augmented models. In addition, some simple extensions involving queueing theory are indicated.

LOAD AND SYSTEM CHARACTERISTICS

The manner in which the system load is characterized is described and important features of EXEC 8 operation and behavior are presented.

Load

The model is intended to describe the steady-state performance of the system under average loading conditions. Such performance is almost assuredly not the same as the performance under a uniform average load characterized by the means of various distributions describing the average load. However, for sake of simplicity, a uniform average load is employed in the subsequent model development. This is a place to start and perhaps as pointed out in References 2 and 4 some

interesting gross performance statistics can be obtained. (While very interesting, it would be extremely expensive to investigate the effects of this assumption experimentally in a meaningful way. It certainly should be pointed out, however, that the resulting model predictions will be optimistic at best. At NTH, it is hoped to look at this question in detail with the aid of a complete simulation model of EXEC 8 now under development.)

Table I presents the average characteristics notation employed to describe the load:

Core quanta

The core-time impact of a task in EXEC 8 is measured by its core quantum, Ψ , which is related to the core quantum time, Q , of a task requiring C blocks of core as follows:

$$\Psi = QC = 2^{(P_c-1)}\Psi_s \quad (1)$$

where:

Ψ_s is an installation specified value. (The NTH EXEC 8 employs $\Psi_s = 512$ block.ms)

P_c is the core priority level of the task. (In EXEC 8, the core priority level of a batch task is fixed at its run priority level [typically 6] while the core priority level of a demand task starts at level 2 and increases with each successive level of the exponential CPU time scheduling algorithm that the task experiences [actual core and CPU priority is in inverse order of level number; typically, the priority level of an interactive task remains at level 2]).

Unfortunately for ease of analysis, the core quantum time, Q , of a task is not elapsed time in core, but rather

TABLE I—Load Characteristics Notation

	Batch	Demand
Average core requirements including non-resident executive functions that must be loaded (in units of 512 word blocks)	C_b	C_d
Average CPU time per task core residence (i.e., per swap)	t_b	t_d
Number of open/active jobs (N_b is an installation parameter and constant under heavy load while n_d describes demand load with the installation fixing an upper bound)	N_b	n_d
Average total CPU time per job	T_b	

is measured in terms of CPU time and channel time charged to the task (concurrent usage is only charged once).

Demand impact control philosophy

EXEC 8 attempts to limit the impact of demand service on system performance by dynamically adjusting system behavior to maintain a statistic referred to here as the demand service ratio, DSR, at an installation specified limit function of the number of active demand users, n_d . The value of this statistic is computed at six second intervals and averaged over the last several minutes of operation. If necessary, EXEC 8 temporarily raises the core priority of the next batch task to insure loading and increases its core quantum, Ψ_b , to correct the measured DSR to the desired DSR during the next six second interval. The DSR is defined as:

$$\text{DSR} = (\text{Core quanta charged to demand} + \text{core-time product of total swap activity}) / (\text{Core quanta charged to both batch and demand} + \text{core-time product of total swap activity}) \quad (2)$$

To further quantify this relation, it is necessary to develop more precise terminology.

EXEC 8 core scheduling algorithm

In terms of core quanta, the EXEC 8 core scheduling algorithm exhibits the following behavior:

- (1) When core conditions change, the highest core priority task ready for load-in is checked for ability to fit in the available space. If possible, space is reserved and a swap-in is initiated. If not, the task type is checked; if a batch task, the next lower priority ready task is checked, etc.; if a demand task, the core scheduler is disengaged until the next core status change. In order to prevent excessive delays, wait times are accumulated on each of the waiting tasks and after too long a wait, core entry is forced by temporarily raising the core priority.
- (2) Once in core, a task is guaranteed of remaining for its full core quantum or until it voluntarily relinquishes core control by entering a terminal I/O or long wait state or unless exceptional system conditions such as I/O buffer full state occur.
- (3) Upon completing its core quantum, a task is considered swappable by a higher core priority

demand task. In the absence of demand task pressure, the system does relatively little swapping. The main cause of swapping in a pure batch environment is dynamic facilities conflicts caused generally by two core resident jobs employing the same system processor (e.g., FORTRAN) resulting in a long wait state which may lead to swapping.

From a uniform load, steady-state modeling viewpoint, the problem is to abstract the important aspects of the fairly complicated core scheduling algorithm and ignore fine structure details such as the controls to deal with excessive waits by tasks with large core requirements.

CPU time scheduling algorithm

Once a task has received core, it is subject to a complex multilevel exponential CPU scheduling algorithm. Fortunately, from a steady-state modeling viewpoint for a limited core configuration such as that of NTH, CPU scheduling has smaller impact on system performance and therefore need not be given as much attention here as core scheduling. For sake of completeness though, the essence of the algorithm is as follows:

- (1) A queue of queues is maintained for both batch and demand type tasks. Each successive queue is a higher level and has associated with it a lower CPU priority and a larger CPU time quantum.
- (2) Both batch and demand tasks start their core quanta at level 2 CPU priority (their interrupt activities are processed at level 1 and they are forced if necessary at level 0). If a task at level 2 relinquishes control of the CPU prior to completion of its CPU time quantum, it is queued for service at level 2 upon completion of whatever I/O action caused it to relinquish control. So long as the task remains at level 2, it receives a new level 2 CPU time quantum each time it receives CPU service. If, however, a task does not relinquish control and runs to the end of its CPU time quantum, it is queued for service at the end of the queue at the next level with a CPU time quantum that is twice as large as it had previously.
- (3) So long as the task remains compute bound (i.e., it does not voluntarily release the CPU), it moves up the priority levels each move resulting in a doubling of CPU time quantum until it reaches level 7. As soon, however, as it voluntarily relinquishes control it is requeued for

service at the end of the level 2 queue with the original level 2 CPU time quantum.

All this of course is subject to the constraints of the core quanta scheduling scheme described previously.

Demand cycle

The definition of the *demand cycle* is needed to quantify the relationships just described. The demand cycle is artificial; i.e., it has no physical counterpart in the system. However, it provides a way of introducing a cyclic time frame into an otherwise steady-state model. If there are n_d average active demand tasks competing for system resources, all other things equal, they will be serviced in cyclic order. The sequential execution of these n_d tasks in combination with sufficient batch tasks to maintain the DSR is termed a demand cycle. The number of batch tasks that are swapped into core and executed during a demand cycle is denoted as n_{bd} .

Demand batch ratio

It will be convenient to parameterize the relationship between n_{bd} and n_d by the concept of a *demand batch ratio* denoted by DBR. The DBR like n_{bd} is an artificial quantity; these quantities do not appear physically in EXEC 8, but appear implicitly as a direct function of DSR. The DBR is the average ratio of CPU time allocated to demand tasks to that allocated to batch tasks. In view of the definitions, the DBR can be computed over a demand cycle as:

$$\text{DBR} = n_d t_d / n_{bd} t_b \quad (3)$$

In terms of defined quantities, DSR defined in (2) can be quantified as:

$$\text{DSR} = \frac{(n_d \Psi_d + 2(n_d C_d S_d + n_{bd} C_b S_b))}{(n_d \Psi_d + (n_{bd} \Psi_b) \text{SF} + 2(n_d C_d S_d + n_{bd} C_b S_b))} \quad (4)$$

where:

SF is an installation-set scale factor. (The value employed by Univac in the NTH EXEC 8 is $\frac{3}{2}$.)

S_b and S_d denote the times to swap the average batch and demand tasks into (or out of) core and are respectively:

$$\begin{aligned} S_b &= T_A + C_b T_F \\ S_d &= T_A + C_d T_F \end{aligned} \quad (5)$$

where:

T_A is the average access time to locate a swap file

and/or system processor on the swapping drum (for the FH 432, $T_A = 4.3ms$).

T_F is the flow time per 512 word block of information from (or to) the swapping drum (for the FH 432, $T_F = 2.13ms$).

Substituting relations from (1) and (3) into (4) and solving for DBR yields:

$$\begin{aligned} \text{DBR} &= \left(\frac{C_d}{C_b} \right) \left[\left(\frac{\text{DSR}}{1 - \text{DSR}} \right) \right. \\ &\quad \left. \times \left(\frac{Q_b}{t_b} \right) \text{SF} - 2 \left(\frac{S_b}{t_b} \right) \right] / \left[\left(\frac{Q_d}{t_d} \right) + 2 \left(\frac{S_d}{t_d} \right) \right] \quad (6) \end{aligned}$$

which corroborates the previous assertion that DBR is a direct function of DSR. For simplicity, DBR is employed in the following formulation to represent the effect of the system control actions. These actions are mechanized in EXEC 8 in terms of DSR, but (6) establishes that the effect can be described in terms of DBR.

BASIC MODEL

The underlying assumptions are justified and a basic model is developed.

Simplifying arguments

In Reference 5, Hellerman and Smith present a simple, but elegant, model for throughput analysis of record processing EDP applications for various physical and logical overlap configurations. Their simplifying assumptions exclude a number of important EDP applications, but interestingly enough cover the case of full swap, buffered time sharing systems. Now EXEC 8 introduces additional complexity through its batch multiprogramming features, but Reference 5 provides interesting insights that serve as the basis of the current model.

In particular, computing the batch and demand swap times from (5) for the NTH average core requirements of: $C_b = 50$ blocks, $C_d = 25$ blocks; yields: $S_b = 111ms$, $S_d = 58ms$. The NTH observed compute time per swap for batch and demand of: $t_b = 160ms$, $t_d = 20ms$, plus the observation that swapped in tasks must also be swapped out, suggests that under heavy demand load the single swapping channel on an 1108 configuration such as that of NTH will be very busy. With the exception of some cycle stealing conflicts, the tasks' compute activity can be overlapped completely by swap activity and with sufficient core buffer space available, the

swapping channel might saturate under heavy demand load. Under swapping saturation, the difference between compute times and swap times would appear to provide more than half of the total possible CPU time per demand cycle to handle EXEC 8 functions.

These simplifying arguments lead to the following assumptions for the basic model:

Basic assumptions

- (1) The swapping channel is saturated.
- (2) All compute time on both batch and demand is completely overlapped by swap time.
- (3) All system overhead is also completely overlapped by the swap time.
- (4) There is sufficient core available for buffering so that the above assumptions are reasonable.

Performance calculations

Assumptions 1, 2, and 3 above allow the total elapsed time for the demand cycle, T_{dc} , to be quantified as the sum of the total demand and batch swap time:

$$\begin{aligned} T_{dc} &= 2(n_d S_a + n_{bd} S_b) \\ &= 2n_d t_a [(S_a/t_a) + \text{DBR}^{-1}(S_b/t_b)] \end{aligned} \quad (7)$$

T_{dc} provides an upper bound to R_d , the system response time to demand users, assuming cyclic service to the n_d users; i.e., $R_d < T_{dc}$.

T_{dc} can also be employed to compute the elapsed processing time for a batch job, ET_b which serves as a lower bound on the expected turnaround time, R_b , for the average batch job. R_b also includes the time spent waiting in the system backlog queue and depending on definition may also include waiting time in a system input queue prior to the backlog queue, a printer queue, and one or more output handling queues prior to delivery back to the user. If there are N_b open batch jobs receiving equal service, the amount of CPU time allocated to a single batch job during a demand cycle is $(n_{bd}/N_b)t_b$. If the requisite CPU time for batch job execution is T_b , the total elapsed time for an average batch job is:

$$R_b \geq ET_b = (T_b N_b / t_b n_{bd}) T_{dc} \quad (8)$$

The total CPU time used during the demand cycle is:

$$\text{CPU}_{dc} = n_d \cdot t_a + n_{bd} \cdot t_b \quad (9)$$

The average CPU utilization can be computed over the

demand cycle as:

$$\begin{aligned} \text{CPU percent} &= \frac{\text{CPU}_{dc}}{T_{dc}} \cdot 100 \\ &= \frac{(1 + \text{DBR}^{-1}) \cdot 100}{2[(S_a/t_a) + (S_b/t_b) \text{DBR}^{-1}]} \end{aligned} \quad (10)$$

Interestingly, (7) and (8) predict that after swapping saturation, demand response time and batch turnaround time will increase linearly with increasing number of demand users n_d . Also, (10) indicates that with swapping saturation in a system controlled to a fixed demand service ratio, the CPU utilization is independent of n_d .

VERIFICATION

The basic model is adjusted to the NTH load and system characteristics and an attempt is made to verify the model against observed system behavior.

System conditions

The load and system parameters presented in Table II have been directly observed in the NTH environment as a part of a detailed measurement study.

TABLE II—NTH Load and System Parameters

$C_b = 50$ blocks	$C_d = 25$ blocks
$N_b = 5$	$n_d = 6$
$T_b = 25$ sec	$\Psi_d = 1000$ block.ms
$\Psi_b = 16000$ block.ms	$t_d = 20$ ms
DSR = .35	

The average batch load characteristics of Table II also agree with those observed by the University of Wisconsin in a recently reported measurement study.⁸

From the values of Table II and (1) and (5), it is possible to compute the intermediate model parameters of Table III:

TABLE III—Intermediate Model Parameters

$S_b \approx 111$ ms	$S_d \approx 58$ ms
$Q_b \approx 320$ ms	$Q_d \approx 40$ ms

TABLE IV—DBR Versus DSR

DBR	DSR
.01	.30
.05	.35
.15	.40

In order to verify the basic model, it only remains to obtain a realistic value for t_b , the CPU time per swap-in (or equivalently per core quantum) for a batch task. The t_d of 20ms reported in Table II is measured directly. There are two pieces of experimental data that permit estimation of t_b :

- (1) The observed ratio of Q_d/t_d is 2, and
- (2) Wisconsin reports in Reference 8, that the average total channel time per batch job is twice the average CPU time. No overlap of individual channel time or CPU time would, as described in the LOAD AND SYSTEM CHARACTERISTICS section, yield a Q_b/t_b of 3 and complete overlap of two channels and the CPU will yield a Q_b/t_b of 1.

For a university environment with its heavy use of system processors that make good use of overlap possibilities it is not unreasonable to expect that the Q_b/t_b ratio will be the same as the experimentally observed Q_d/t_d ratio of 2.

On the basis of this argument, a t_b of 160ms is employed in the sequel.

Performance predictions

Solution of (6) for the given parameter values yields a DBR of .05 for a DSR of .35. In experimental studies at NTH, DBR values lower than .01 have been observed with an n_d of 6, but, as is later developed, appreciably larger DBRs are necessary to support the swapping saturation assumption. Thus in the sequel, calculations are performed for DBR values given in Table IV with corresponding DSR values.

Solution of Equations (7), (8), and (10) as a function of DBR for the parameter values of Tables II and III yields Table V:

TABLE V—Performance of Basic Model

DBR	$T_{dc}(\text{sec})$	$ET_b(\text{sec})$	CPU Percent
	($n_d = 10$)		
.01	29	181	70
.05	7	210	63
.15	3	282	51

Measurements at NTH for the average load characteristics of Table II and with a DBR of less than .01 at an n_d of 6 indicated an average CPU utilization of 50 percent, an average batch turnaround of 5 min., and an average demand response time of 38 seconds. The basic model predictions of Table V indicate an appreciably lower T_{dc} which causes a lower ET_b and a higher CPU percent than that measured. Moreover other measurements indicated that the swapping channel was approximately 60 percent busy rather than the 100 percent assumed by the model under similar loading conditions. This lack of agreement prompts investigation of the validity of the assumptions supporting this basic model.

AUGMENTED MODEL

The swapping saturation assumption itself is subject to question, but analysis of this is presented in the next section. This section explores the question of available core, modifies the basic model, and predicts system performance from the resulting augmented model.

Required core

One approach to understanding the effect of available core on model performance is to analyze the amount of core necessary to maintain swapping saturation. This is done by first considering the integrated core-time demand over a demand cycle, CTD_{dc} . If EQ_d and EQ_b are respectively the effective elapsed times demand and batch tasks remain in core when swapped in for their core quanta, then average demand and batch jobs tie up C_d and C_b blocks of core for: $2S_d + EQ_d$ and $2S_b + EQ_b$ ms respectively. Thus in one demand cycle the integrated core-time demand is:

$$CTD_{dc} = n_d C_d (2S_d + EQ_d) + n_b C_b (2S_b + EQ_b) \quad (11)$$

block-ms, and the minimum average core requirement to maintain a demand cycle of T_{dc} is:

$$C_{min} = CTD_{dc} / T_{dc}$$

$$= \frac{C_d \left[2 \left(\frac{S_d}{t_d} \right) + \left(\frac{EQ_d}{t_d} \right) \right] + \frac{C_b}{\text{DBR}} \left[2 \left(\frac{S_b}{t_b} \right) + \left(\frac{EQ_b}{t_b} \right) \right]}{2 \left[\left(\frac{S_d}{t_d} \right) + \text{DBR}^{-1} \left(\frac{S_b}{t_b} \right) \right]} \quad (12)$$

The problem in analyzing (12) lies in the determination of EQ_d and EQ_b . As explained previously,

TABLE VI— C_{\min} for Varying DBR and EQ/Q

		(Units of 512 Word Blocks)		
DBR \	EQ_d/Q_d	1	3/2	2
	EQ_b/Q_b	1	2	3
.01		119	188	257
.05		107	167	227
.15		87	134	180

Q_d and Q_b are well defined quantities for average demand and batch jobs. Unfortunately, it is non-trivial to relate the elapsed core time, EQ , to the core quantum time, Q , which is counted only when CPU and/or channel activity is occurring for that task. The following arguments are pertinent:

- (1) For all batch task swap-ins, but the swap-in where the task terminates, $EQ_b > Q_b$. The elapsed residence time of the last swap-in may be less than Q_b because the task will voluntarily give up core control upon terminating. Inasmuch as T_b/t_b , the number of swap-ins for the average batch job, is greater than 100 and this job will involve approximately three sequential tasks (compilation, collection, and execution), these end effects will have little effect on average behavior. Hence $EQ_b > Q_b$.
- (2) For the average demand task swap-in, there are end effects more frequently because the interactive tasks will voluntarily give up core control upon entering terminal wait state. In the EXEC 8 environment however, the average user employs batch oriented processors (compilers and the collector) for a large proportion of the swap-ins. Also because the demand CPU time quantum is more closely related to the demand core quantum time than for the case of batch, the relation in (13) is almost certainly true on the average:

$$1 < EQ_d/Q_d < EQ_b/Q_b \quad (13)$$

- (3) In order to estimate the magnitude of the ratios in (13), it should be noted that each time the task is eligible to receive CPU or channel attention it will on the average have to wait $\frac{1}{2}$ (in the case of one competitor) or more of the length of time it needs to employ the resource. For these reasons EQ_d/Q_d is probably greater than $\frac{3}{2}$ and EQ_b/Q_b , because of the increased employment of channel resources by batch tasks, is almost assuredly greater than that.

Since these ratios appear to be very much a function of what is happening in the model, they are carried as parameters in the augmented model development and verification.

Equation (12) is solved to yield the values in Table VI as a function of DBR and the EQ/Q ratios.

For certain parameter combinations, the C_{\min} required to sustain saturated swapping is greater than C_A , the actual number of core blocks available for user tasks at a particular installation. In practice, C_{\min} must be in multiples of C_d (or C_b) blocks. Thus the numbers in Table VI must be increased at least to C_A or the next higher multiple of C_d (whichever is larger). This modified C_{\min} is denoted as C'_{\min} .

There is one other usage of core that must be taken into account. For each possible active demand user, space must be reserved for line buffers, switch list information, etc., in the executive buffer pool area (EXPOOL). Each possible user requires the reservation of approximately 0.8 of a 512 word block of core store. Thus the C_A employed to represent the number of available user blocks is really a function of n_d :

$$C_A = C_{AO} - .8n_d \quad (14)$$

(In the NTH configuration, there are a total of 256 blocks of which $C_{AO} = 130$ and typically the system has been generated for a maximum n_d of 6 yielding an effective C_A of 125 blocks for user tasks.) In the following model performance predictions, the NTH C_{AO} of 130 is used.

As with C_{\min} , when postulating an average model, C_A only has meaning as a multiple of C_d (or C_b) blocks. Thus all use of C_A must be in terms of C'_A where C'_A is the C_A defined in (14) reduced to the next lower multiple of C_d .

Model development

The *minimum* effect of having too little core to buffer the tasks under swapping saturation will be to increase T_{dc} by an amount proportional to the ratio between C'_{\min} and C'_A . Such an increase in T_{dc} will cause a reduction in CPU percent by the inverse ratio.

Thus for the case of swapping saturation limited by available core, the basic model Equations (7), (8), and (10) are augmented as follows:

$$T'_{dc} = (C'_{\min}/C'_A) T_{dc} \quad (15)$$

$$ET'_b = (C'_{\min}/C'_A) ET_b \quad (16)$$

$$\text{CPU}' \text{ percent} = (C'_A/C'_{\min}) \text{CPU percent} \quad (17)$$

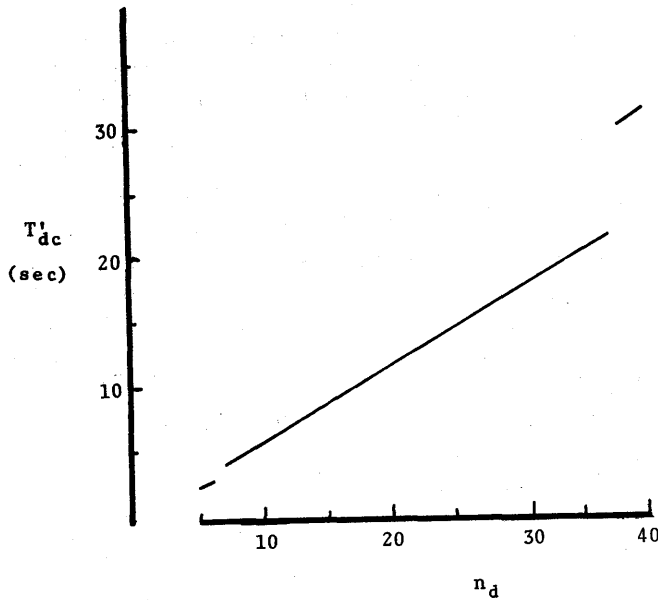


Figure 1—Response Time Upper Bound (T'_{dc}) vs. Number of Demand Users (n_d)
 DBR = .10, $EQ_d/Q_d = 3/2$, $EQ_b/Q_b = 2$

Model performance predictions

The ratio of C'_A/C'_{min} can also be employed to predict the percentage of time that the swapping channel will be busy for the core limited case.

Solutions of (15) and (17) for varying DBR, n_d , and EQ/Q yields the results in Tables VIII and IX. The n_d values of 6, 20, and 30 are selected because NTH currently has an n_d of 6, plans to expand to an n_d of 20 before acquiring more core, and is interested in the impact of an even greater number of terminals. DBR values in the range .01 to .15 are employed to investigate the effect of a DSR in the range .30 to .40 as in Table IV.

For ease of sensitivity analysis, Figures 1 and 2

TABLE VII— C'_A/C'_{min} (for Two EQ/Q Couplets)*
 Swapping Channel Busy Percentage Varying DBR and n_d

DBR \ n_d	$n_d \leq 6$	$6 < n_d \leq 37$	$39 \leq n_d < 69$
.01	(63, 46)	(50, 36)	(38, 27)
.05	(72, 50)	(57, 40)	(43, 30)
.15	(84, 63)	(67, 50)	(50, 38)

* In each table entry (A, B), A is computed for the $\{EQ_d/Q_d, EQ_b/Q_b\}$ of $\{3/2, 2\}$ and B for $\{2, 3\}$.

TABLE VIII— T'_{dc} (for Two EQ/Q Couplets)*
 Varying DBR and n_d
 (Units of Seconds)

DBR \ n_d	6	20	30
.01	(28, 38)	(116, 159)	(173, 239)
.05	(6, 8)	(23, 34)	(35, 50)
.15	(2, 3)	(9, 12)	(14, 18)

present plots of T'_{dc} and CPU' percent respectively for an EQ/Q couplet of $\{3/2, 2\}$ and a DBR of .10 as a function of n_d .

Model verification

At first analysis the performance predictions in Tables VIII and IX and Figures 1 and 2 appear to agree reasonably well with NTH experience. For example on December 16, 1970, in approximately 10 hours of EXEC 8 operation there were 725 batch jobs involving 3325 batch tasks; six demand terminals were available and a total of 58 demand user sessions were conducted involving 1437 demand service requests. For the average job characteristics reported in Table II, the average CPU utilization was 50 percent, average

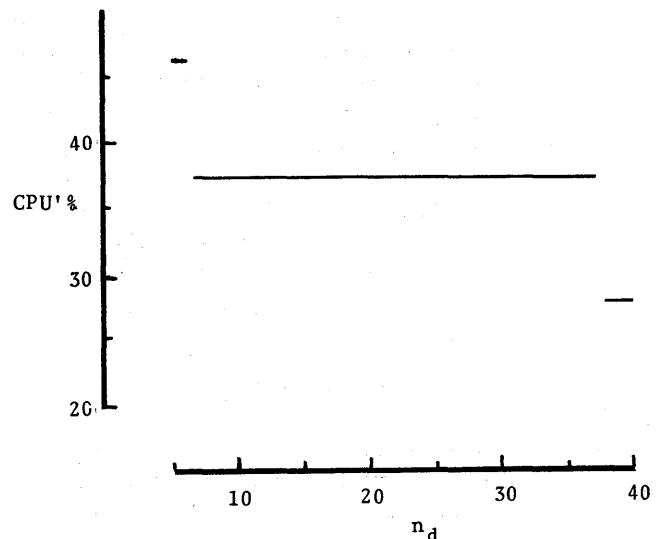


Figure 2—CPU Utilization (CPU' Percent) vs. Number of Demand Users (n_d)

DBR = .10, $EQ_d/Q_d = 3/2$, $EQ_b/Q_b = 2$

batch turnaround (last card in to job terminate) was 5 min., and average demand response time was 38 seconds.

This amount of demand usage corresponds to an average DBR of less than .01 for which the model predicts (assuming an EQ/Q couplet of $\{\frac{3}{2}, 2\}$): CPU' percent = 44, $T'_{dc} = 28$ sec., $ET'_b = 3$ min. During one hour of heavier demand usage, there were 285 demand service requests with an average t_d of 50ms corresponding to an average DBR of .01 and the observed CPU utilization was 40 percent. Moreover the previously reported swapping channel busy percentage of 60 percent under similar conditions agrees well with the prediction of 63 percent reported in Table VII for an EQ/Q couplet of $\{\frac{3}{2}, 2\}$.

On an absolute basis, considering the gross simplification of the model, this sort of agreement must be considered to be quite good. It isn't until the swapping saturation analysis presented in the next section is reviewed, that it becomes apparent that this agreement is probably due more to system and model insensitivity than to true representation.

What can be said with some certainty, however, is that the relative predictions as a function of n_d and DBR are realistic. The relative sensitivity of T'_{dc} to n_d is not particularly surprising, but the sensitivity of the CPU' percent to n_d is of some real concern. It has already been reported in Reference 8 and observed experimentally at NTH that each new demand user subtracts approximately .04 from CPU utilization prior to swapping saturation. Table IX and Figure 2 indicate that, even after swapping saturation, allowing addition of new demand users can cause significant reduction of system thruput due to reduction of available user core.

ANALYSIS AND EXTENSIONS

The augmented model of the preceding section is a direct result of analyzing the basic assumption concerning sufficient core. The relative success of this

TABLE IX—CPU' Percent (for Two EQ/Q Couplets)*

Varying DBR and n_d			
DBR \ n_d	$n_d \leq 6$	$6 < n_d \leq 37$	$38 \leq n_d < 69$
.01	(44, 32)	(35, 25)	(26, 19)
.05	(45, 31)	(36, 25)	(27, 19)
.15	(42, 32)	(34, 25)	(25, 19)

* See Table VII.

analysis suggests that the swapping saturation assumption should also be investigated. Simple queuing theory extensions are indicated and conclusions are drawn.

Swapping saturation

As indicated previously, in the absence of demand usage there is relatively little swapping. (How little is indicated by results presented in Reference 8, where during one day's all batch operation under EXEC 8 involving 1249 batch jobs, there were 10935 extra swap-ins from the swap-file or approximately 8 swap-ins per average job.)

If this ratio is preserved in the batch/demand environment, of the $T_b/t_b (= 156)$ swap-ins per batch job suggested by the model, more than 90 percent would have to be caused by pressure from waiting higher priority demand tasks. During a demand cycle then, the ratio of n_d/n_{bd} must be greater than .9 to force swap-out of batch tasks thereby causing swapping saturation. Combined with the NTH t_b/t_d ratio of 8, (3) suggests that DBR must be greater than .10 independent of n_d to cause swapping saturation.

n_d enters into the analysis in terms of the number of users necessary to have a realistically large response time at this relatively large DBR. If for example, an average user can maintain an effective demand rate of 2 interactions/minute requiring t_d ms of CPU time and having EQ ms of elapsed core time impact, then (15) shows that swapping saturation can only be maintained by approximately 38 users for an EQ/Q couplet of $\{\frac{3}{2}, 2\}$ and 33 users for a couplet of $\{2, 3\}$.

Thus the model cannot really be verified on the basis of NTH experience with its n_d of 6.

Configuration sensitivity

What can be ascertained for the NTH 1108 configuration however, is the potential effect of making a planned move to twenty demand terminals.

For example, currently one day's usage with light demand activity from 6 possible users and without swapping saturation yields a CPU utilization of 50 percent and a batch turnaround of 5 minutes. Figure 2 indicates that heavy usage from 30 or more demand users could cause swapping saturation with a CPU' percent reduction of twenty to fifty percent. Certainly the results of Table VI suggest that NTH could obtain a significant performance increase through the addition of more core.

The dependence of EXEC 8 performance on core had, of course, been suspected prior to the model develop-

ment reported here. What this work does however, is to give a means of quantifying the cost/benefit analysis. Moreover because of the nature of the uniform average load assumption, it is safe to assert that the absolute model performance predictions are optimistic; i.e., the CPU' percent is high and T'_{dc} is low. It is asserted however, that with the exception of the unspecified EQ/Q load dependence, the relative model performance predictions are reasonable.

Extensions

There are a number of possibilities for improving the existing model; for example, the EQ/Q ratio could be related to other characteristics describing the system, load, and, in particular, potential conflict situations on the channels. In light of other problems inherent in the uniform load and swapping saturation assumptions however, such modifications do not appear to be particularly fruitful at this time.

The upper bound relation of T'_{dc} to R_d , the system response time for demand users, and the lower bound relation of ET'_b to R_b , the processing turnaround time for batch users, suggests that some simple queueing theory additions could make the nature of these inequalities much better understood. As pointed out in Reference 3, one of the biggest problems in applying queueing theory to general multiprogramming-time sharing systems such as EXEC 8 is to determine effective mean service rates (and, of course, service time distributions) for the various types of service offered by the system.

Interestingly, the model developed here provides mean service rates for demand interactions and batch job service requests. For example, a rough first approximation to determining improved estimates for R_d follows directly from Reference 3 for a finite population model with n_d sources with Poisson service request distributions about some mean rate λ . The service model could be single server with an exponential service time distribution with effective rate μ_{de} . Since each service request requires $2S_d$ ms of swapping service, μ_{de} is:

$$\mu_{de} = P_d / 2S_d \quad (18)$$

where:

P_d is the probability that the swapping drum is serving demand users.

P_d is computed in (19) from the relations of the model:

$$P_d = 2n_d S_d / T'_{dc} \quad (19)$$

The prediction of R_b could also be improved by a

simple queueing model with the service rate determined in a similar manner. Completion of the queueing theory analysis is best deferred until better experimental data is available for verification.

CONCLUSIONS

As indicated in the INTRODUCTION, the initial goals for this model development were twofold; namely: (1) to predict steady-state performance as a function of configuration and workload, and (2) to serve as a focus for detailed study of EXEC 8 design and behavior.

Most of the discussion to this point has concentrated on the first of these goals. A concluding summary of this discussion is that the presented model gives surprisingly accurate predictions of system behavior outside its designed range. There are at present no measurements available for verification of model validity under swapping saturation. However, the trend predictions of the model are interesting and certainly could be useful if an EXEC 8 configuration causing swapping saturation were being studied. From a more general modeling viewpoint, perhaps the greatest value of this work is as a well documented example of a limiting resource modeling situation.

The model and associated study are perhaps most significant in relation to the second goal. That is, the model served as a focus for more general study and measurement of EXEC 8 and provided a means for quantifying completely undocumented, yet very important, topics such as core quanta and DSR control.

REFERENCES

- 1 A L SCHERR
Analysis of time shared computer systems—simulation of TSS
MIT Research Monograph No 36 MIT Press 1967
- 2 R R FENICHEL A J GROSSMAN
An analytic model of multiprogrammed computing
AFIPS Proceedings 1969 SJCC Vol 34 pp 717-721
- 3 J E SHEMER D W HEYING
Performance modeling and empirical measurements in a system designed for batch and time sharing users
AFIPS Proceedings 1969 FJCC Vol 36 pp 17-26
- 4 B W ARDEN D BOETTNER
Measurement and performance of a multiprogramming system
Proceedings of the ACM 2nd Symposium on Operating System Principles ACM October 1969 pp 130-146
- 5 H HELLERMAN H J SMITH
Throughput analysis of some idealized input, output, and compute overlap configurations
Computing Surveys Vol 2 No 2 June 1970 pp 111-118

6 G ESTRIN L KLEINROCK

*Measures, models, and measurements for time shared
computer utilities*

Proceedings of ACM National Conference 1967 Vol 22
pp 85-96

7 S R KIMBLETON C G MOORE

*A probabilistic framework for system performance
evaluation*

Proceedings of the ACM SIGOPS Workshop on System
Performance Evaluation Harvard University Cambridge
Massachusetts April 5-7 1971

8 M DRAPER

1108 EXEC-8 performance evaluation and scheduling

Presented to Fall 1970 Univac Users Association (USE)
Meeting University of Wisconsin Computing Center
Madison Wisconsin October 1970

Throughput measurement using a synthetic job stream*

by DAVID C. WOOD and ERNEST H. FORMAN

The MITRE Corporation
McLean, Virginia

INTRODUCTION

Managers of computer facilities frequently need to be able to measure the throughput of their installations. Throughput is based on the amount of work a system can perform in a given time in a particular environment, and is usually measured as the reciprocal of the running time of selected jobs. The throughput of a computer system is a function of the hardware configuration, the operating system software and the workload characteristics. Once the characteristics of the workload are identified, the configuration and operating system can be optimized in terms of throughput for that environment. Improvements resulting from system changes can be evaluated using throughput measurements.

With a given workload and system configuration, CPU utilization is a crude measure of relative throughput. More accurate results are obtainable by careful experimentation with benchmarks which are representative of the workload. A benchmark usually refers to a single job, such as a matrix inversion which may be typical of certain scientific applications. Benchmarks are frequently used to evaluate different computer systems prior to their selection.¹ With a multiprogramming computer system, the throughput depends on the job mix, the scheduling algorithm, and many parameters in the operating system. A job stream is a collection of independent jobs which can be used to determine the relative throughput of a multiprogramming system based on the time taken to execute all the jobs.

A job stream could be gathered from the actual workload by using a sampling procedure and verifying the resulting job stream by comparing features with a total workload. Such a study at the University of Iowa has been reported.² This is not feasible in all installa-

tions. The following difficulties have been experienced in assembling and using a job stream composed of actual jobs:

- users of a service facility are reluctant to supply programs, data bases and operating instructions
- security considerations prevent many jobs from being included in the job stream
- it is difficult to closely match the overall characteristics of the jobstream to the workload characteristics without being able to select from a large number of jobs since the characteristics of each job are fixed
- it is extravagant on storage space to duplicate large data bases
- new releases of the operating system and changes in catalogued procedures make it difficult to keep complex jobs viable.

An alternative is to create synthetic jobs reflecting characteristics of the workload. This paper describes experience in defining the characteristics of a workload and generating a synthetic job stream based on the workload characteristics. There is little published work in either of these areas, and there are many practical difficulties to be solved. The synthetic job stream has been validated by comparison with a job stream comprised of actual jobs, and with the use of a hardware monitor. It has been used to measure the throughput of alternative hardware configurations and alternative software options.

CHARACTERIZATION OF WORKLOAD

There are no established standards for describing characteristics of a workload. Characteristics which are defined must be obtainable quantitatively for the workload, and since they are to be used as a basis for assembling a job stream, they should be easily trans-

* This work was carried out by the MITRE Corporation under contract to the Defense Communications Agency (Contract Number F19628-71-C-0002).

latable into components of the job stream. The characteristics introduced in this paper relate to the IBM 360 series operating under OS/MVT; similar terms could be defined for other machines. Most of the characteristics can be obtained by an analysis of system accounting data.

Characteristics of the workload are defined in terms of the resource requirements imposed on the system. Resources considered are:

- Core requirement
- CPU utilization
- I/O channel activity
- Peripheral device utilization

Core requirement is easily measured for a real memory, in contrast to a virtual memory machine. The distribution of time spent in various region sizes can be calculated. CPU utilization can be expressed as the percentage of time the CPU is active. Accounting statistics almost always give a measure of I/O activity, but in some machines, such as the IBM 360, the measure may be inadequate for throughput studies. The measure given with the Systems Management Facility (SMF) of OS is the number of data transfers, not the quantity of data transferred. To resolve this deficiency, a hardware monitor has been used to measure the average channel activity per data transfer.

Although CPU and I/O channel utilization are important, they are not easily translated into jobs which will produce this activity. This can best be achieved by considering the ratio of I/O channel time to CPU time, both as a total for the workload, and as a distribution. For example, in a particular installation, short jobs may generally be dominated by I/O time, whereas long jobs are more often CPU bound. The utilization of tape units and exchangeable disk units can be measured in terms of the percentage of time these units are assigned to some job. The amount of activity while these units are assigned to a job and of public devices such as work disk packs is already included in the I/O channel utilization. The demand on printers, card readers and punches is easily calculated in terms of lines of print or number of cards.

DESIGN OF JOB STREAM SPECIFICATIONS

The process of translating the workload characteristics into a synthetic job stream has two steps. First, the specifications of the individual jobs are determined so that they match the workload. Second, the synthetic jobs themselves are generated with the specifications determined previously.

Designing the job stream specifications based on the workload characteristics is the more difficult task. The required running time of the job stream is determined from the average job time and the accuracy desired. The number of jobs required can then be deduced. Several different types of synthetic jobs may be used; for example, jobs requiring tape or disk or neither. Core size, CPU times, and I/O times can be defined for each job in order to match the overall workload distributions for each characteristic, but it is not obvious how the various characteristics should be correlated for individual jobs. For example, although the probability of larger I/O time increases as the CPU time of a job increases, this is not always the case.

An approach to the correlation problem is to base the specifications of the synthetic jobs on actual jobs. A sampling technique can be used to select jobs in the actual workload, and the sampling procedure can be verified by comparing the characteristics identified previously. However, instead of obtaining the actual jobs, with the attendant practical difficulties noted earlier, only their characteristics are used. The core requirement, CPU and I/O time, peripheral requirements, printer output, et cetera, of each job can be used as the specifications for creating a synthetic job which looks identical.

GENERATION OF SYNTHETIC JOB STREAM

The synthetic job used in this study is based on the type of program suggested by Buchholz.³ A listing of the program, which is written in PL/I, appears in the Appendix. Parameters in the program are adjusted to vary the CPU time, I/O time and number of lines of printer output. Region size and tape and disk requirements of each job are determined by suitable specification of the job control language (JCL). Most synthetic jobs consist of compile, link edit and go steps of the PL/I program, but in those cases where a small region size or short CPU time is required, only the go step of a previously compiled and link edited program is executed.

The PL/I program performs the following tasks:

- (a) Creates a data set of master records (data set DD name is MASGEN)
- (b) Processes records from this data set (data set DD name is MASTER)
- (c) Exercises a compute kernel a number of times for each record processed
- (d) Outputs from one to three data sets after each record has been processed (data set DD names are OUT1, OUT2, OUT3. If less than three output data sets are required, OUT2 and/or OUT3 may be declared "DUMMY")

- (e) Prints some number of lines after each record is processed.

The content of the master records, the type of calculations within the compute kernel, the content of the output records and the content of the printed lines are to a large extent arbitrarily selected. Some effort was made in these selections to include a variety of operations. The execution time of the compute kernel is approximately 30 milliseconds on a 360/50.

There are five input parameters to the synthetic PL/I job;

- (a) NMASTER: The number of master records to be created.
- (b) R: The number of records to be processed from the master file ($R \leq \text{NMASTER}$).
- (c) N: The number of times the compute kernel is to be executed.
- (d) L: The number of lines to be printed.
- (e) LPR: The number of lines to be printed out for each record processed until the total number of lines of printed output is equal to L.

The four data sets in the PL/I program (MASGEN-MASTER, OUT1, OUT2, OUT3) are assigned unit and volume designations in the JCL according to the tape and user-disk requirements for the job. Blocking factors for these data sets are also specified in the JCL.

The I/O time reported by the System Management Facility (SMF) routine under OS/MVT (release 18) is based on the number of execute channel programs (EXCPs). The larger the blocking factor, the smaller the number of job EXCPs for a given number of records processed. By increasing both the number of records processed and the blocking factors, it is possible to keep the number of EXCPs for a job constant while increasing the amount of actual I/O taking place. There are many variations that can be applied to the synthetic job. For example, the amount of CPU time used between each I/O operation can, instead of being fixed, be made a random variable subject to the constraint that the total job CPU time is as specified. The degree of sophistication of the synthetic job must be weighed against the resulting complexity of the experiments and the effect on the validity of the results.

MEASUREMENT PROCESS

The running time of the job stream has been calculated as the elapsed time from starting the card reader until all processing is completed. The order in which the jobs are arranged may have some effect on

TABLE I—Activity Monitored with Hardware Monitor (Minutes)

	Synthetic	Representative
CPU Active	32.4	34.2
Supervisor (OS/HASP) Time	22.7	19.0
Selector Channel 1 Busy	12.4	26.6
Selector Channel 2 Busy	13.5	30.4
Selector Channel 4 Busy	14.6	35.1

the running time. It is preferable to avoid having a long job running last when there are no other jobs to multi-program with it. The job stream should be run in two or three different orders and the average time calculated. If the job stream is long enough, the variations will be relatively small. Maximum differences rarely exceeding seven percent have been experienced with a three hour job stream where the average running time of a job is seven minutes.

It is important to start each run with a clean system to ensure consistency; for example, work disk packs should be scratched. Any delays which may be caused by operator intervention, such as mounting user disk packs, should be avoided as far as possible.

COMPARISON OF ACTUAL AND SYNTHETIC JOB STREAMS

A potential disadvantage of a synthetic job stream is that it may not be representative of the actual workload because of the lack of characteristics not explicitly designed into it. The validity of using a synthetic job stream to measure throughput has been confirmed by comparison with a representative job stream. The representative job stream, running for about three hours on a 360/50, was comprised of actual jobs selected from the workload so as to reflect the workload characteristics defined previously. A synthetic job stream was created with each job having the same characteristics (CPU time, I/O time, region size, lines of output, and tape and disk requirements) as the corresponding representative job.

Both job streams have been run on a 360/50 and 360/65, both with and without HASP. The elapsed time for the synthetic job stream runs varied between 71 and 76 percent of the elapsed time for the corresponding representative job stream. Table I shows the activity data obtained with a COMRESS DYNAPROBE hardware monitor. The percentage of time the selector channels were busy with the synthetic job stream running was approximately half that for the representative job stream. This is because the synthetic job

TABLE II—HASP Performance

	360/50I		360/65J	
	Synthetic	Representative	Synthetic	Representative
Time with HASP	126	166	54	76
Time without HASP	141	194	82	109
% (HASP/Non-HASP)	89.4	85.6	65.8	69.7

stream was tuned with the number of EXCPs used to represent I/O activity. More data must be transferred per EXCP in order that the selector channels are kept suitably busy. This can be done by using larger blocking factors in the synthetic jobs. The other activities recorded by the hardware monitor showed close agreement between the two job streams, as can be seen from the figures given in Table I.

Both job streams have been used to determine the relative throughput of a 360/65 with 1024K bytes of core in relation to a 360/50 with 512K bytes of core. The performance of HASP on each of these configurations has also been evaluated using both job streams.

The throughput factor for the 360/65 in terms of the 360/50 was measured with the representative job stream to be 2.18. The throughput factor measured with the synthetic job stream was 2.33. This difference can be attributed to the fact that the synthetic job stream is less I/O bound and was therefore able to make better use of the faster CPU of the 360/65.

The improvement in performance obtained with HASP on each of the two machines as measured by the representative and synthetic job streams is shown in Table II. These measures are in quite close agreement. The deficiency of I/O in the synthetic job stream had little effect in this case since the major performance benefit of HASP is derived from "SYSIN" and "SYSOUT": I/O which did not differ substantially between the two jobs streams.

SUBSEQUENT RESULTS WITH A SYNTHETIC JOB STREAM

Further use has been made of a synthetic job stream to determine throughput factors of three different IBM 360 configurations, to determine relative CPU speeds (as reported by the accounting routine) of these configurations, and to evaluate the effect of a hardware and operating system change on one of these con-

figurations. In order to equitably charge users for attended time on each of the three different configurations, a 360/50-I (512K bytes of core), a 360/65-I (512K bytes of core), and a 360/65-J (1024K bytes of core), a measure of the relative throughput of these machines was needed. A synthetic job stream was used to obtain this measure. In order to assure a true representation of the I/O activity in the synthetic job stream, hardware monitor measurements were made of actual workload channel activity for a two week period and the I/O of the synthetic job stream was made to accurately reflect the true workload I/O. The resulting synthetic job stream had more I/O than the original synthetic job stream and more even than the representative job stream. This indicates that the representative job stream itself had too little actual I/O as compared to the workload. The throughput factor of the 360/65-J in terms of the 360/50-I as measured with the new synthetic job stream was 1.75. This decrease from the throughput factors measured with the representative and original synthetic job streams clearly illustrates the strong dependence of computer performance on the I/O workload characteristics.

The CPU times reported by the accounting routine for synthetic job stream runs on each configuration were compared and CPU factors derived using the 360/50-I as a reference. Table III contains the CPU time for runs on each machine and the CPU factors. It is interesting to note that the CPU of the model 65 with the smaller core appears to be faster than the 65 with the megabyte core. Although it is known that CPU time for a given job does vary from run to run on a given 360, it is statistically significant that all 25 jobs of the synthetic job stream were charged with less CPU time on the 360/65-I than on the 360/65-J. Two primary factors why the 360 accounting time varies are improper distribution of CPU time for I/O interrupt processing and cycle stealing. These factors are directly proportional to the amount of multiprogramming and I/O activity respectively, both of which are greater on the 360/65-J machine.

The managers of the computer installation felt that the throughput of one of the machines had decreased following a hardware change and a transition to a newer release of the operating system. The hardware

TABLE III—CPU Times and Factors

Machine	CPU Time	CPU Factor
360/50-I	3334	1.00
360/65-J	1070	3.12
360/65-I	823	4.05

change involved the replacement of a selector channel which was used for tape I/O, by multiplexor sub-channels. Data from the hardware monitor had shown that the selector channel that was removed and the multiplexor channel had very little usage so that a cost savings could be realized with no predicted degradation in performance.

The synthetic job stream previously run on this machine was run in the new environment. Contrary to there being a degradation in performance, the synthetic job stream ran in less time thus showing an increase in throughput for the same workload characteristics.

CONCLUSIONS

The PL/I synthetic program described can be used as the basis for a job stream reflecting the workload characteristics which are considered important in affecting throughput: CPU utilization, I/O channel activity, core requirement, printer output, and tape and disk requirements. Experimentation with a synthetic job stream designed with these characteristics matching those of a job stream comprised of actual jobs has shown comparable results. This supports the choice of

characteristics on which the synthetic job stream is based. A hardware monitor is invaluable in determining I/O channel activity in the absence of such a measure for 360 computers.

A synthetic job stream is a more practical tool than a job stream composed of actual jobs for measuring throughput. The synthetic job stream is easier to assemble and does not require extensive data bases. It is particularly advantageous where the regular workload is classified. Synthetic jobs are not dependent on compilers or data management systems and therefore do not require maintenance because of system changes in those areas.

REFERENCES

- 1 E O JOSLIN
Computer selection
Addison-Wesley 1968
- 2 W L SHOPE K L KASHMARAK
J W INGRAM W F DECKER
System performance study
Proceedings SHARE XXIV March 1970 pp 568-659
- 3 W BUCHHOLZ
A synthetic job for measuring system performance
IBM Systems Journal Vol 8 No 4 1969

APPENDIX

PL/I LISTING OF SYNTHETIC PROGRAM

```

SYN1: PROCEDURE OPTIONS (MAIN);
      DECLARE 1 MASTER_REC ALIGNED STATIC,
              2 MASTER_KEY CHARACTER(12),
              2 MASTER_SUM BINARY FIXED(31),
              2 MASTER_CHECK BINARY FIXED(31),
              2 MASTER_DATA (5) CHARACTER(12),
              INTKEY PICTURE'(6)9',
              CHECK BINARY FIXED(31) INITIAL(0),
      PARS FILE INPUT,
      MASGEN FILE RECORD OUTPUT,
      MASTER FILE RECORD INPUT,
      OUT1 FILE RECORD OUTPUT,
      OUT2 FILE RECORD OUTPUT,
      OUT3 FILE RECORD OUTPUT,
              LINES BINARY FIXED(31) INITIAL(1),
              NREPS BINARY FIXED(31) INITIAL(0),
              N BINARY FIXED(31),
              L BINARY FIXED(31),
              NMASTER BINARY FIXED(31),
              LPR BINARY FIXED(31),
              R BINARY FIXED(31);

/* GET R, N, L, NMASTER AND LPR */
      GET FILE (PARMS) DATA;
      PUT DATA(R,N,L,NMASTER,LPR); PUT SKIP(5);

/* CREATE NMASTER MASTER RECORDS */
      OPEN FILE(MASGEN); DO J=1 TO NMASTER;
      CHECK = CHECK + J;
      INTKEY = J;
      MASTER_KEY='000000' || INTKEY;
      MASTER_DATA = '000000' || INTKEY;
      MASTER_CHECK = CHECK;
      MASTER_SUM = 0;
      WRITE FILE(MASGEN) FROM (MASTER_REC);
      END; CLOSE FILE (MASGEN);

      OPEN FILE (MASTER);
      OPEN FILE (OUT1);
      OPEN FILE (OUT2);
      OPEN FILE (OUT3);
DO L1 = 1 TO R;
      READ: READ FILE (MASTER) INTO (MASTER_REC);
/* EXECUTE KERNEL N/R + 1 TIMES PER RECORD OR 1 TIME PER RECORD UNTIL
   N TOTAL REPETITIONS */
      DO L2 = 1 TO ((N-R)/R + 1 + 1) WHILE (NREPS<N);
      DECLARE IX BINARY FIXED(31) INITIAL(13571),
              IY BINARY FIXED(31),RN DECIMAL FLOAT;
      ON FIXEDOVERFLOW;
      IY = IX*65539;
      IF IY < 0 THEN IY=IY +2147483647+1;
      RN = IY;
      RN = RN * .4656613E-9;
      RN = RN * 10.;
      IX = IY;
      INTKEY=SQRT(RN);MASTER_DATA(1)='000001' || INTKEY;
      INTKEY=EXP(RN) ;MASTER_DATA(2)='000002' || INTKEY;
      INTKEY = (1+ 3*3.14)/LOG(RN);MASTER_DATA(3)=
              '000003' || INTKEY;
      INTKEY =L1; MASTER_DATA(4)='000004' || INTKEY;
      INTKEY =L2; MASTER_DATA(5)='000005' || INTKEY;
      MASTER_SUM=MASTER_SUM+1;
      MASTER_CHECK = L1;
      NREPS = NREPS + 1;
      END;

      WRITE: WRITE FILE (OUT1) FROM (MASTER_REC);
      WRITE FILE (OUT2) FROM (MASTER_REC);
      WRITE FILE (OUT3) FROM (MASTER_REC);

/* PRINT LINES LPR LINES PER RECORD WHILE LINES <=L */
      DO K=1 TO LPR WHILE (LINES<=L);
      PUT SKIP DATA(L1,L2,LINES,NREPS);
      LINES= LINES+1; END;

END;
END SYN1;

```


A feedback queueing model for an interactive computer system

by GISAKU NAKAMURA

*Musashino Electrical Communication Laboratory, NTT
Musashinoshi, Tokyo, Japan*

INTRODUCTION

Recently considerable effort has been directed to the development of computer systems that are able to serve a large number of users in an interactive manner. The model of interactive computer system is described by stating what a single user does during an elementary operation at his console, the "interaction." Roughly stated, an interaction consists of the user requesting and then receiving service from the computer system. The events usually forming an interaction are: the user's thinking, typing at his remote console, waiting for a response from the computer system, and finally watching output. These interactions are repeated until the user finds the desired output. The number of interactions depends on the contents of a job which is processed by the computer system and on the goodness of program which is processed by the user in each interaction. Since this number fluctuates stochastically, it may be considered as a random variable.

The turnaround time is defined as the time interval between the generation of the first request and the reception of the final service from the computer system. Thus, the complete service for a job is made during the turnaround time. For users the turnaround time is one of the most important characteristics of the computer system. There are some characteristics of the interactive computer system which are of operational importance. These characteristics are:

- (a) The service time, which is the duration of time required to complete the service for a request in an interaction;
- (b) The response time, which is the time interval between the generation of a request in an interaction and the reception of service from the computer system in the same interaction;
- (c) The think time, which is the time interval between the reception of service in an interaction

and the generation of a request in the next interaction; and

- (d) The interaction time, which is the time interval during which an interaction is completed, that is, the sum of the response time and the think time.

In this paper, a simple mathematical model of the operation is proposed for an interactive computer system, and some analyses are made for the turnaround time, the response time, the interaction time, and the number of jobs in the computer system.

MOTIVATION OF THE ANALYSIS

There are various analytical models for time sharing computer systems, which are extensively surveyed by J. M. McKinney.¹ Most of the models have been constructed for the purpose of estimating the response characteristics in each interaction. Some typical models are the round-robin model, the multi-level foreground background model, and the external priority model. The analyses of the models have well explained the servicing behavior for requests in each interaction, and the probability distribution of response time and related characteristics have usually been obtained in postulated time sharing environments. These models, however, are not suitable to explain the servicing behavior for jobs through the complete turnaround time, namely, the models are too complex to be used in estimating the characteristics in connection with the entire sequence of interactions between users and the computer.

Unfortunately, little work has been carried out in analyzing the overall servicing behavior of interactive computer systems from a mathematical point of view, at least to the author's knowledge. The motivation for lack of the analysis is directed toward constructing an

analytical model by which entire interactions between users and the computer are described in a suitable way.

ANALYTICAL MODEL

Assumptions

To construct an analytical model for the interactive computer system, we will make some assumptions on the arrival to the system, on the stochastic behavior of the service times and the think times, and on the number of interactions during the turnaround time. First, it is natural to assume that the sequence of job arrivals constitutes a Poisson process with a constant arrival rate, because jobs arrive at the computer system independently each other. In fact, this assumption has been empirically justified in various situations, and has been adopted in almost all queueing models of time sharing systems.¹ Second, the service times should depend on a scheduling algorithm by which requests in each interaction are processed. But, in a very rough estimation, it may be considered that they are mutually independent and identically distributed. We assume the exponential distribution of service times in the usual way. This is also demonstrated by practical data observed by A. L. Scherr² with compatible time sharing systems. Similar considerations can be made on the think times. It is evident that a user's think time in an interaction is independent from his think time in another interaction as well as from another user's think time. Scherr's observation also supports this assumption as an approximation. Here, it is noted that the exponential distribution assumptions of the service times and the think times make the model tractable. Finally, the number of interactions between a user and the computer fluctuates stochastically and may be considered as a random variable. We assume that the probability of having another interaction of a job is independent of the number of interactions preceding it.

Description of the model

Suppose that jobs arrive at the computer system in accordance with a Poisson process with density λ . Denote by τ_n ($n=1, 2, 3, \dots$) the arrival epoch of the n th job. Then, the interarrival times $\tau_{n+1} - \tau_n$ ($n=1, 2, 3, \dots$) are identically distributed, mutually independent random variables with distribution function

$$A(t) = \begin{cases} 1 - e^{-\lambda t} & \text{if } t \geq 0, \\ 0 & \text{if } t < 0. \end{cases}$$

The jobs are served by a single processor in order of

arrival. The processor is idle if and only if there is no job in the computer system. The service times are supposed to be identically distributed, mutually independent random variables with exponential distribution function

$$H(t) = \begin{cases} 1 - e^{-\mu t} & \text{if } t \geq 0, \\ 0 & \text{if } t < 0. \end{cases}$$

After being served, each job either returns the computer system with probability γ , requesting further service, or goes away permanently with probability $1-\gamma$. The event that a job returns is independent of any other event involved and, in particular, independent of the number of its previous returns. In the case that a job returns the computer system, some delay (the think time) is required before the job joins the queue again. The think times are supposed to be identically distributed, mutually independent random variables with exponential distribution function

$$G(t) = \begin{cases} 1 - e^{-\nu t} & \text{if } t \geq 0, \\ 0 & \text{if } t < 0. \end{cases}$$

It is supposed that the newly arrived jobs and the returned jobs are equally treated in the computer system. Thus, all jobs are served in order of arrival. The distribution function of service times for the returned jobs is supposed to be equal to that for the newly arrived jobs.

The model described is a kind of single-server queueing model with feedback. A single-server queueing model with feedback has been investigated by L. Takács.³ In his model, the Poisson arrival is assumed and the feedback rate γ is defined in the same way. The service times are supposed to be identically distributed, mutually independent random variables with general distribution function. From this point of view,

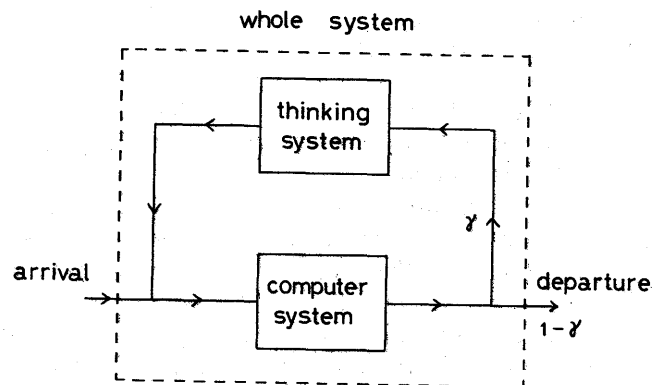


Figure 1—The queueing model with feedback

Takács' model is more general than ours. However, immediate returns are required when jobs join the queue again, while some delay is involved in our model.

The model is well described by introducing a virtual thinking system in which infinitely many servers are furnished. Let the whole system consist of the computer system and the thinking system, as is shown in Figure 1. Jobs that arrive at the whole system enter the computer system and join the queue. The jobs are served by a single processor in order of arrival. After being served, each job either immediately enters the thinking system with probability γ or departs from the whole system with probability $1-\gamma$. Since infinitely many servers are furnished in the thinking system, there is no queue in it. Therefore, the service in the thinking system is immediately commenced when a job enters the thinking system. The distribution function of service times in the thinking system is given by $G(t)$. After being served, each job immediately enters the computer system and joins the queue. The cycles from computer system to thinking system are repeated until the job departs from the whole system. Then, the probability with which a job has n returns is given by

$$r_n = \gamma^n (1-\gamma), \quad n=0, 1, 2, \dots \quad (1)$$

ANALYSIS

The mean number of jobs in the system

Assume that the whole system is in statistical equilibrium. Let ξ be the random variable representing the number of jobs in the computer system, and let η be the random variable representing the number of jobs in the thinking system. Denote by $p(i, j)$ the probability with which $\xi=i$ and $\eta=j$. Then, it is obtained that

$$\lambda p(0, 0) = \mu(1-\gamma)p(1, 0), \quad (2)$$

$$(\lambda + j\nu)p(0, j) = \mu(1-\gamma)p(1, j) + \mu\gamma p(1, j-1), \quad j > 0 \quad (3)$$

$$(\lambda + \mu)p(i, 0) = \lambda p(i-1, 0) + \mu(1-\gamma)p(i+1, 0) + \nu p(i-1, 1), \quad i > 0, \quad (4)$$

$$(\lambda + \mu + j\nu)p(i, j) = \lambda p(i-1, j) + \mu(1-\gamma)p(i+1, j) + \mu\gamma p(i+1, j-1) + (j+1)\nu p(i-1, j+1), \quad i > 0, j > 0. \quad (5)$$

The probability generating function of ξ and η is defined by

$$P(x, y) = E[x^\xi y^\eta] = \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} p(i, j) x^i y^j,$$

where E represents the mathematical expectation.

Multiplying (2), (3), (4), and (5) by $1, y^j, x^i,$ and $x^i y^j$ respectively and summing them, it is derived that

$$\begin{aligned} & \nu(y-x)P_y(x, y) + [\lambda(1-x) \\ & \quad + \mu\{1 - (1-\gamma)/x - \gamma y/x\}]P(x, y) \\ & = \mu\{1 - (1-\gamma)/x - \gamma y/x\}P(0, y), \end{aligned} \quad (6)$$

where we put

$$P_y(x, y) = dP(x, y)/dy.$$

Let L_2 be the mean number of jobs in the thinking system. Then,

$$L_2 = \sum_{j=1}^{\infty} j \left(\sum_{i=0}^{\infty} p(i, j) \right) = P_y(1, 1). \quad (7)$$

L_2 is easily found in the following way. Putting $x=1$ in (6), it is obtained that

$$\begin{aligned} & \nu(y-1)P_y(1, y) + \mu\gamma(1-y)P(1, y) \\ & = \mu\gamma(1-y)P(0, y). \end{aligned} \quad (8)$$

Then, by differentiating both sides of (8) with respect to y and then putting $y=1$, it is obtained that

$$\nu P_y(1, 1) = \mu\gamma\{P(1, 1) - P(0, 1)\}. \quad (9)$$

Substituting (7) and $P(1, 1) = 1$ in (9), it is obtained that

$$L_2 = \mu\gamma\{1 - P(0, 1)\}/\nu. \quad (10)$$

Here, $P(0, 1)$ is found as follows. Put $y=x$ in (6). Then,

$$\begin{aligned} & (1-x)\{\mu(1-\gamma) - \lambda x\}P(x, x) \\ & = (1-x)\mu(1-\gamma)P(0, x). \end{aligned} \quad (11)$$

By differentiating both sides of (11) with respect to x and then putting $x=1$, it is obtained that

$$P(0, 1) = 1 - \lambda/(1-\gamma)\mu. \quad (12)$$

Hence, by substituting (12) in (10), it is obtained that

$$L_2 = \lambda\gamma/(1-\gamma)\nu. \quad (13)$$

Next, let L_1 be the mean number of jobs in the computer system. Then,

$$L_1 = \sum_{i=1}^{\infty} i \left(\sum_{j=0}^{\infty} p(i, j) \right) = P_x(1, 1) \quad (14)$$

where we put

$$P_x(1, 1) = P_x(x, y) \Big|_{x=1, y=1} = dP(x, y)/dx \Big|_{x=1, y=1}.$$

Since

$$dP(x, x)/dx = P_x(x, x) + P_y(x, x),$$

it is derived that

$$dP(x, x)/dx \Big|_{x=1} = L_1 + L_2. \quad (15)$$

Differentiating both sides of (11) two times with respect to x and then putting $x=1$, it is obtained that

$$\lambda - \{\mu(1-\gamma) - \lambda\}(L_1 + L_2) = -\mu(1-\gamma)P_y(0, 1). \quad (16)$$

Here, it is noted that

$$dP(0, x)/dx = P_y(0, x).$$

Thus, (16) yields

$$L_1 + L_2 = \rho/(1-\rho) + P_y(0, 1)/(1-\rho), \quad (17)$$

where we put

$$\rho = \lambda/(1-\gamma)\mu.$$

ρ is the utilization factor of the computer system, because

$$\lambda(1 + \gamma + \gamma^2 + \dots) = \lambda/(1-\gamma)$$

is the arrival rate and $1/\mu$ is the mean service time.⁴ The second term in the right hand side of (17) is the mean number of jobs in the thinking system, under the condition that there is no job in the computer system. Then, if the state in the thinking system is stochastically independent of that in the computer system, L_2 may be given by

$$L_2 = P_y(1, 1) = P_y(0, 1)/(1-\rho).$$

Under the assumption of this stochastic independence, L_1 may be given by

$$L_1 = \rho/(1-\rho). \quad (18)$$

In the following section we will prove the stochastic independence between the states in the computer system and in the thinking system.

Method of finding $P(x, y)$

Assume that the states in the computer system and in the thinking system are stochastically independent. Then, the probability generating function $P(x, y)$ is factorized as

$$P(x, y) = P(x, 1)P(1, y). \quad (19)$$

We will show that $P(x, y)$ given by (19) becomes the solution of (6) if $P(x, 1)$ and $P(1, y)$ are suitably chosen.

Denote by M/M/1 the single-server queueing system with a Poisson arrival and exponential service times.⁵ It is known that the mean number of jobs in the system M/M/1 is given by $\rho/(1-\rho)$, where ρ is the utilization factor of the system. This formula coincides with the first term in the right hand side of (17). Thus, it is suspected that the computer system forms the system M/M/1 from the queueing point of view. Since the

probability generating function of the number of jobs in the system M/M/1 is given by $(1-\rho)/(1-\rho x)$,⁴ we put

$$P(x, 1) = (1-\rho)/(1-\rho x). \quad (20)$$

Next, denote by M/M/ ∞ the infinitely many-server queueing system with a Poisson arrival and exponential service times.⁵ It is also known that the mean number of jobs in the system M/M/ ∞ is given by λ'/ν , where λ' is the arrival rate and $1/\nu$ is the mean service time. Now, under the assumption of the stochastic independence, the second term in the right hand side of (17) is written as

$$\lambda'/\nu = \{\lambda\gamma/(1-\gamma)\}/\nu,$$

where

$$\lambda' = \lambda\gamma/(1-\gamma) = \lambda(\gamma + \gamma^2 + \gamma^3 + \dots)$$

is the arrival rate of the thinking system. Thus, it is suspected that the thinking system forms the system M/M/ ∞ from the queueing point of view. Since the probability generating function of the number of jobs in the system M/M/ ∞ is given by $\exp\{-\lambda'/\nu(1-x)\}$,⁴ we put

$$P(1, y) = e^{-\beta(1-y)} \quad (21)$$

where

$$\beta = \lambda\gamma/(1-\gamma)\nu.$$

Substituting (20) and (21) in (19), it is obtained that

$$P(x, y) = P(x, 1)P(1, y) = \frac{(1-\rho)e^{-\beta(1-y)}}{1-\rho x}. \quad (22)$$

To show that $P(x, y)$ given by (22) is the solution of (6), put

$$A_1 = \nu(y-x)P_y(x, y),$$

$$A_2 = [\lambda(1-x) + \mu\{1 - (1-\gamma)/x - \gamma y/x\}]P(x, y),$$

$$A_3 = \mu\{1 - (1-\gamma)/x - \gamma y/x\}P(0, y).$$

Then, it is sufficient to prove that

$$A_1 + A_2 - A_3 = 0.$$

Since A_1 and A_3 are calculated as

$$A_1 = \nu\beta(y-x)P(x, y),$$

$$A_3 = \mu\{1 - (1-\gamma)/x - \gamma y/x\}(1-\rho x)P(x, y),$$

it is derived that

$$\begin{aligned} (A_1 + A_2 - A_3)/P(x, y) &= \lambda\gamma(y-x)/(1-\gamma) + \lambda(1-x) \\ &\quad + \lambda\{x - (1-\gamma) - \gamma y\}/(1-\gamma) \\ &= 0. \end{aligned}$$

Thus, (22) becomes the solution of (6) and our conjecture that the states in the computer system and in the thinking system are stochastically independent is justified.

In the above analysis, it is shown that the computer system and the thinking system form the systems M/M/1 and M/M/∞ respectively. This fact can be intuitively explained in the following way. Assume that the computer system forms the system M/M/1 from the queueing point of view. It is known that the output process of the system M/M/1 is a Poisson process.⁶ It is also known that the probabilistic selection of jobs from a Poisson process results in a Poisson process.⁷ Hence, the input process of the thinking system becomes a Poisson process and then the thinking system forms the system M/M/∞ from the queueing point of view. Since the output process of the system M/M/∞ is a Poisson process,⁶ and the aggregation of several independent Poisson processes results in a Poisson process,⁷ the input process of the computer system becomes a Poisson process. Thus, the computer system forms the system M/M/1 from the queueing point of view, which is our first assumption. Therefore, no contradiction is derived, and our assumption is justified. This intuitive argument will be used later.

The mean turnaround time

The turnaround time is defined as the time interval between the generation of the first request and the reception of the final service from the computer system. In other words, the turnaround time is the time interval during which a job stays in the whole system. The mean turnaround time is one of the most important characteristics for users.

Denote by $T(n)$, ($n=0, 1, 2, \dots$), the mean turnaround time for jobs with n returns. Since the probability with which a job has n returns is given by (1), the arrival rate for jobs with n returns is written as

$$\lambda(n) = \lambda r_n = \lambda \gamma^n (1 - \gamma), \quad n = 0, 1, 2, \dots \quad (23)$$

Let $L_1(n)$ be the mean number of jobs with n returns in the computer system, and let $L_2(n)$ be that in the thinking system. Then, by applying Little's theorem to the whole system,⁸ it is obtained that

$$T(n) = \{L_1(n) + L_2(n)\} / \lambda(n), \quad n = 0, 1, 2, \dots \quad (24)$$

We will find $L_1(n)$ and $L_2(n)$.

Let $L_1(n, k)$ be the mean number of jobs with n returns in the computer system, under the condition that those jobs already have k ($k \leq n$) returns. And let $L_2(n, k)$ be that in the thinking system under the same

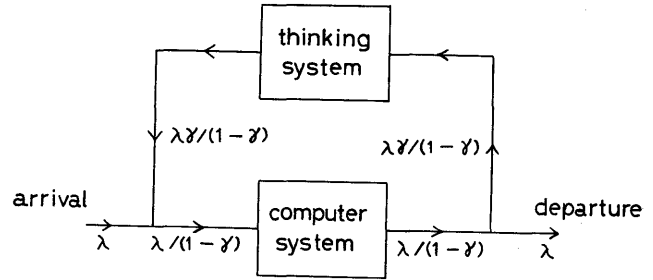


Figure 2—The aspect of flows in the system

condition. Then, it is evident that

$$L_1(n) = \sum_{k=0}^n L_1(n, k), \quad (25)$$

$$L_2(n) = \sum_{k=0}^{n-1} L_2(n, k). \quad (26)$$

In statistical equilibrium, the input rate in any system is equal to the output rate in the same system. Now, define

$$u_1(n, k) = L_1(n, k) / L_1,$$

$$u_2(n, k) = L_2(n, k) / L_2.$$

Then, by equating the input rate of jobs with n returns in the computer system and the output rate, it is obtained that

$$\lambda \gamma^n (1 - \gamma) = \{\lambda / (1 - \gamma)\} u_1(n, n), \quad (27)$$

$$\begin{aligned} & \{\lambda \gamma / (1 - \gamma)\} u_2(n, k - 1) \\ & = \{\lambda / (1 - \gamma)\} u_1(n, k), \quad n \geq k \geq 1. \end{aligned} \quad (28)$$

Here, it is noted that the input rates in the computer system and in the thinking system are $\lambda / (1 - \gamma)$ and $\lambda \gamma / (1 - \gamma)$ respectively, as is shown in Figure 2. From (27) and (28), it is derived that

$$u_1(n, n) = \gamma^n (1 - \gamma)^2, \quad (29)$$

$$\gamma u_2(n, k - 1) = u_1(n, k), \quad n \geq k \geq 1. \quad (30)$$

Similarly, by equating the input rate of jobs with n returns in the thinking system and the output rate, it is obtained that

$$\{\lambda / (1 - \gamma)\} u_1(n, k) = \{\lambda \gamma / (1 - \gamma)\} u_2(n, k), \quad n > k \geq 0. \quad (31)$$

From (31), it is derived that

$$u_1(n, k) = \gamma u_2(n, k), \quad n > k \geq 0. \quad (32)$$

Using (29), (30), and (32), $u_1(n, k)$ and $u_2(n, k)$ are

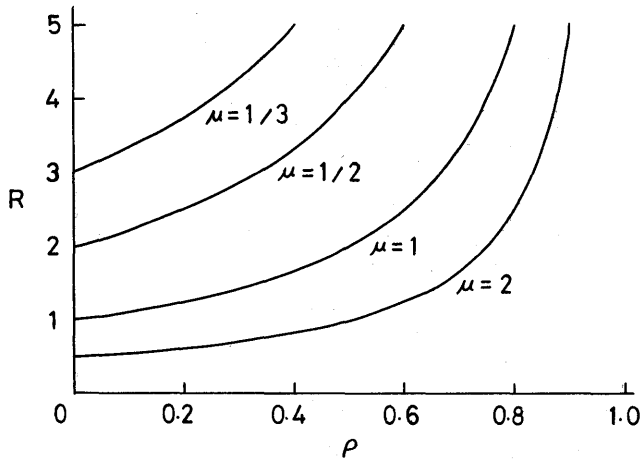


Figure 3—The mean response time R

given by

$$u_1(n, k) = \gamma^n(1-\gamma)^2, \quad n \geq k \geq 0,$$

$$u_2(n, k) = \gamma^{n-1}(1-\gamma)^2, \quad n > k \geq 0.$$

Then, by the definitions of $u_1(n, k)$ and $u_2(n, k)$, it is obtained that

$$L_1(n, k) = \gamma^n(1-\gamma)^2 L_1, \quad (33)$$

$$L_2(n, k) = \gamma^{n-1}(1-\gamma)^2 L_2. \quad (34)$$

Here, it is noted that $L_1(n, k)$ and $L_2(n, k)$ do not depend on k . By substituting (33), (34) in (25), (26) respectively, it is obtained that

$$L_1(n) = (n+1)\gamma^n(1-\gamma)^2 L_1, \quad (35)$$

$$L_2(n) = n\gamma^{n-1}(1-\gamma)^2 L_2. \quad (36)$$

Then, by using (23), (24), (35), and (36), the mean turnaround time $T(n)$ for jobs with n returns is given by

$$T(n) = \frac{(n+1)L_1}{\lambda/(1-\gamma)} + \frac{nL_2}{\lambda\gamma/(1-\gamma)}. \quad (37)$$

Now, we will consider the meaning of (37). Let R be the mean response time of the interactive computer system, and let K be the mean think time. It is shown that the input rate in the computer system is $\lambda/(1-\gamma)$, and the mean number of jobs in that system is denoted by L_1 . Then, by applying Little's theorem to the computer system,³ it is obtained that

$$R = \frac{L_1}{\lambda/(1-\gamma)}. \quad (38)$$

Similarly, the input rate in the thinking system is $\lambda\gamma/(1-\gamma)$ and the mean number of jobs in that system is L_2 , then, by applying Little's theorem to the thinking

system,

$$K = \frac{L_2}{\lambda\gamma/(1-\gamma)} = 1/\nu. \quad (39)$$

Hence, (37) is written as

$$T(n) = R + n(R+K). \quad (40)$$

Here, $(R+K)$ is the mean interaction time for the interactive computer system. Thus, the mean turnaround time for jobs with n returns is the sum of the mean response time and n times the mean interaction time. R , K , and $T(n)$ are shown in Figures 3, 4, and 5 respectively. In the case that a job has no return, of course, the mean turnaround time coincides with the mean response time.

Finally, the mean turnaround time for any job, regardless of the number of its returns, is given by

$$T = \sum_{n=0}^{\infty} r_n T(n) = R + (1-\gamma)(R+K) \sum_{n=1}^{\infty} n\gamma^n$$

$$= R + \{\gamma/(1-\gamma)\}(R+K), \quad (41)$$

which coincides with $(L_1+L_2)/\lambda$.

Extension to the interactive computer system with multiple processors

It is easy to extend the previous analysis to the interactive computer system with multiple processors. Suppose that there are s processors in the computer system. Then, the computer system forms the system

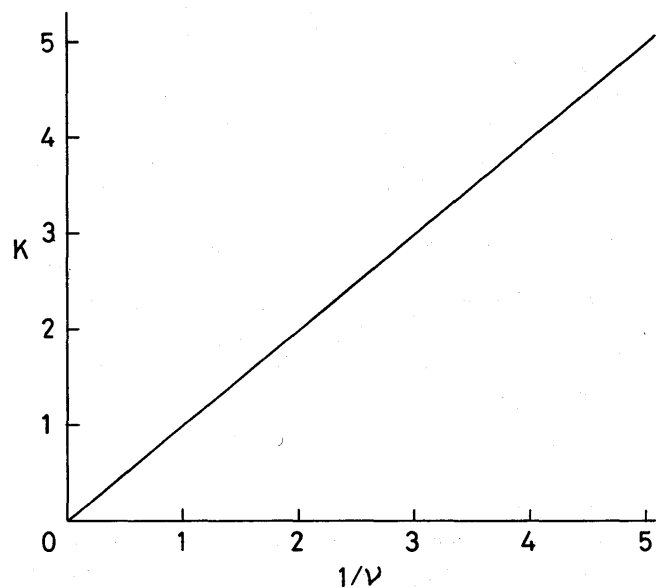


Figure 4—The mean think time K

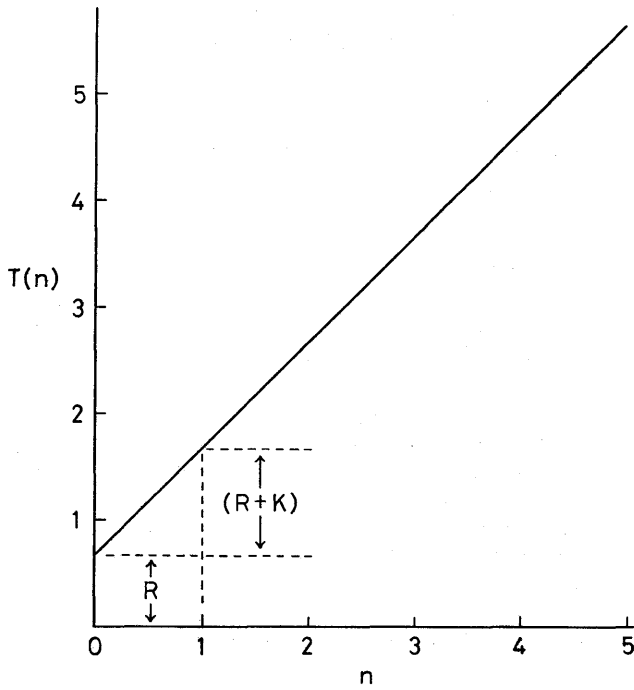


Figure 5—The mean turnaround time $T(n)$

M/M/s from the queueing point of view, where M/M/s means the s servers queueing system with a Poisson arrival and exponential service times.⁵ It is known that the output process of the system M/M/s is a Poisson process.⁶ Then, the intuitive argument given for the analysis of the computer system with a single processor is entirely applied to the analysis of the computer system with s processors. The probability generating function of the number of jobs in the system M/M/s is given by⁴

$$p_0 \left(\sum_{k=0}^{s-1} (\rho x)^k / k! + \sum_{k=s}^{\infty} (\rho x)^k / s! s^{k-n} \right),$$

where p_0 is the probability with which there is no job in the system M/M/s. p_0 is calculated by

$$p_0 = 1 / \left(\sum_{k=0}^{s-1} \rho^k / k! + \rho^s / (s-1)! (s-\rho) \right). \quad (42)$$

Then, if we use

$$P(x, 1) = p_0 \left(\sum_{k=0}^{s-1} (\rho x)^k / k! + \sum_{k=s}^{\infty} (\rho x)^k / s! s^{k-n} \right) \quad (43)$$

instead of (20), the probability generating function $P(x, y)$ is factorized as

$$P(x, y) = P(x, 1)P(1, y)$$

where $P(1, y)$ is given by (21). In this case, the mean

number of jobs in the computer system is given by⁴

$$L_1 = \frac{\rho^{s+1}}{(s-1)! \sum_{k=0}^s (\rho^k / k!) \{(s-k)^2 - k\}}. \quad (44)$$

Then, the previous results (37)-(41) still hold for the interactive computer system with s processors.

DISCUSSION

As an analytical model for the interactive computer system, the paper has proposed a feedback queueing model in which some delay is required before jobs join the queue again. From the analysis of the model, the mean turnaround time is related to the mean response time and the mean think time in a very simple way. Although the validity of this simple relation largely depends on the exponential distribution assumptions for the service times and the think times, it is considered that the result obtained is a good approximation of actual behavior. In fact, Equation (40) which gives the relation can be intuitively justified and can be empirically recognized.

The exponential distribution assumption for the service times is frequently adopted in various queueing models. This is due to the tractability of models as well as to the reasonability of the assumption. Sometimes we may be interested in queueing models with non-exponential distribution assumption. These models usually become hardly tractable in a theoretical way, when they are slightly complicated. However, it is well known that the adoption of the exponential distribution assumption causes results to be on the safe side.

SUMMARY

The paper has proposed a simple mathematical model of the operation for an interactive computer system with a single processor, and has presented some characteristics of the system, such as the mean turnaround time, the mean interaction time, etc. From the queueing point of view, the proposed model is a kind of single-server queueing model with feedback. But, unlike the usual queueing models with feedback, the proposed model requires some delay when a job returns the queueing system. This delay represents the user's think time in the interactive computer system.

The model is well described by introducing a virtual thinking system in which infinitely many servers are furnished. Thus, the whole system consists of the computer system and the thinking system. Here, the user's thinking is represented by the service in the thinking system. The analysis is first made for the

mean number of jobs in the thinking system and for that in the computer system, under the assumptions of a Poisson arrival and exponential service times. Then, the probability generating function of the number of jobs is derived by solving a differential equation. It is shown that the states in the computer system and in the thinking system are stochastically independent, and the computer system and the thinking system form the systems $M/M/1$, $M/M/\infty$ respectively.

ACKNOWLEDGMENTS

The author would like to thank Dr. N. Ikeno, Mr. K. Naemura, and Mr. Y. Yoshida for numerous discussions and suggestions which aided in the preparation of this paper.

REFERENCES

- 1 J M MC KINNEY
A survey of analytical time-sharing models
Computing Surveys Vol 1 No 2 1969
- 2 A L SCHERR
An analysis of time-shared computer systems
Research Monograph No 36 MIT Cambridge
Massachusetts
- 3 L TAKÁCS
A single-server queue with feedback
Bell System Technical Journal Vol 42 No 2 1963
- 4 T L SAATY
Elements of queueing theory
McGraw-Hill New York Toronto London 1961
- 5 D G KENDALL
Stochastic processes occurring in the theory of queues and their analysis by the method of the imbedded Markov chain
Annals of Mathematical Statistics Vol 24 No 3 1953
- 6 P J BURKE
The output of a queueing system
Operations Research Vol 4 No 6 1956
- 7 R W CONWAY W L MAXWELL L W MILLER
Theory of scheduling Chap 8
Addison-Wesley Reading Massachusetts Palo Alto
London Don Mills Ontario 1967
- 8 J D C LITTLE
A proof for the queueing formula: $L = \lambda W$
Operations Research Vol 9 No 3 1961

Alcoa Picturephone Remote Information System (APRIS)

by M. L. COLEMAN, K. W. HINKELMAN, and W. J. KOLECHTA

Aluminum Company of America
Pittsburgh, Pennsylvania

OBJECTIVES AND DESIGN PHILOSOPHY

The objective of the Alcoa Picturephone* Remote Information System (APRIS) is to give to Alcoa executives the capability of using their Picturephones to retrieve information from the corporate computer data base.

The primary design criterion was ease of use. Other management information systems, in an effort to be as powerful as possible, sacrificed simplicity and thus made themselves unsuitable for the personal use of the executive. Experience with these systems has shown that it is unreasonable to expect a busy executive to learn the complex procedures necessary to operate them. In fact it is undesirable, since the job of an executive is to make decisions; anything which interferes with this process, no matter how technologically intriguing, cannot be tolerated.

APRIS's solution to the conflict between ease of use and power was to provide an information center to interpret and respond to the executive's requests for information. Rather than provide just a tool, the goal was to provide a service: the service of better access to information.

APRIS does not require, or even allow, the executive to make retrievals based on complex boolean functions. Rather, by having him press buttons on his Touch-Tone phone, it lets him step through pages of display, one at a time, displaying an index whenever it is necessary to choose between several alternatives. (The complete user guide for the system is shown in Figure 1.) The information center has the responsibility for creating these display pages in response to the executive's demands for information. They can use any techniques available to gather information: existing management information systems (with their complex and powerful logic), independent programs to extract and format the data from the data base used in the daily data processing applications, or manual entry using hardcopy sources.

* Picturephone and Touch-Tone are registered trademarks of the Bell System.

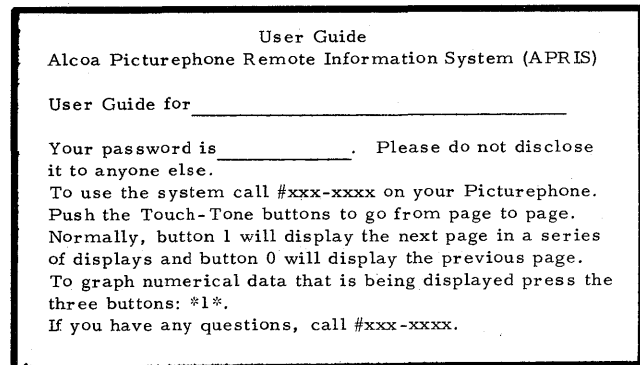


Figure 1—User guide for APRIS

In addition, the information center has a monitor which displays the pages that the executive is seeing and provides audio contact so that the executive can make requests of the information center pertaining to the current data base and the information center can manipulate the display if the executive so desires.

HARDWARE

The current hardware configuration necessary to support Picturephone access at Alcoa is shown in Figure 2. Two lines are presently installed. One, an "intercom" line, allows access from the information center. The other line is connected to the general exchange to allow access from any Picturephone in the calling area.

Each of the lines is connected through a Bell 305 Data Display Set to a 2701 attached to an IBM 360/65 computer. The 305 data set converts Touch-Tone signals from the user to digital codes which are interpretable by the computer and also converts digital codes produced by the computer into video scan lines which are displayable on the Picturephone. The total cost for this configuration, including the 2701 is approximately \$1600 per month. Each additional Picturephone display station costs \$189 a month with exchange service or \$70 a month on the intercom line. A break-

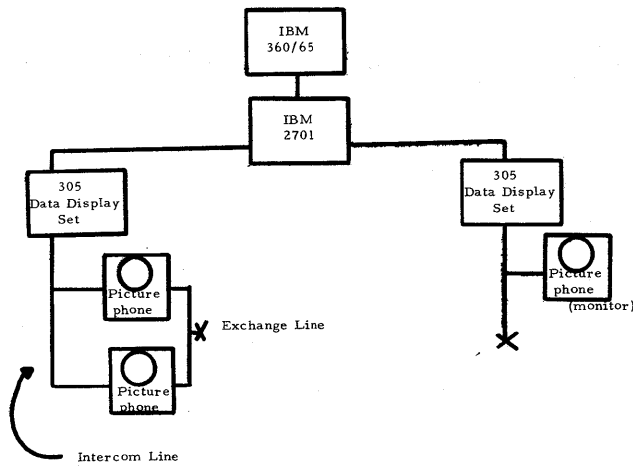


Figure 2—Hardware to support Picturephone access

down of these costs is contained in Figure 3. (These rates are based upon Bell of Pennsylvania tariffs and will vary in other states.)

SECURITY

A tight, multi-level security system is integral to APRIS. To gain access, the proper password must be entered. Each display page is tagged as being public, private, or semi-private giving the capability to restrict the dissemination of confidential data. Data is main-

Basic System

QTY	ITEM	MONTHLY COSTS*
1	2701 with 2 Type III adapters	\$ 550.
2	305 Data Display Sets	550.
1	Picturephone Intercom Circuit	35.
3	Picturephones	210.
2	Business lines with Picturephone service	238.
1	Key service control unit	12.50
	Total	\$1595.50

Additional Terminals

1	Picturephone Display Set	\$ 70.
1	Business line with Picturephone service	119.
	Total	\$ 189.

*Costs are based on Bell of Pennsylvania tariffs and will vary from state to state.

Figure 3—Monthly costs

tained on disk storage in an encrypted form and is not decrypted for display unless all security access requirements are met.

SYSTEM PROGRAMMING

In designing the system it was desired that it be flexible and easy to code. This required the use of a high level language. However, it was also necessary that the system occupy as small an amount of core as possible since it would be resident in the computer the entire day. This required the use of assembly language coding. Both objectives were satisfied by writing and debugging the system in PL/I and then, when the program logic was correct, recoding it in BAL using the PL/I coding as a guide.

Both systems are still in use, the BAL for general use and the PL/I to develop and check out modifications and expansions to the system. Both make use of reentrant code and support multiple, simultaneous Picturephone access.

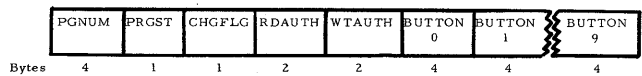
DATA STRUCTURE

Each display page is stored on the disk as a 534 byte record consisting of a 50 byte header followed by the 484 character display page, 22 lines of 22 characters each.

The format of the header is shown in Figure 4.

GRAPHICS

A limited graphic capability has been provided. By pressing a three button code, an executive can have



- PGNUM is the number of the page.
- PRGST is a code which tells the system how to graph the data appearing on that page.
- CHGFLG is used as a flag during alteration of the page.
- RDAUTH is the read authorization field. The first four bits indicate whether the page is public, private, semi-private, or information center. The remainder is a code specifying the access number of the owner if the page is private, or a pointer to a list of access numbers if the page is semi-private.
- WTAUTH is the write authorization field.
- BUTTON 0 - BUTTON 9

are 10 fields each of which contain the page number of the page which will be displayed after the corresponding Touch-Tone button is pressed. A 1 in a field means that the user's initial page (as defined in a table) is displayed when that button is pressed. A zero in a field means that that button is undefined.

Figure 4—Header format

numerical data displayed as a bar graph. Both positive and negative values can be graphed. The system labels the *x*-axis but there is no room on the screen to indicate values for the *y*-axis. A push of a button, however, returns the corresponding numerical display. While austere, the graphics serve to effectively highlight trends and thus significantly improve the usefulness of the system.

PICTUREPHONE vs. CRTs

The display capabilities of the Picturephone are 22 lines of 22 characters with the first and last lines non-useable. Many system analysts feel this is too small to display useful information and thus would prefer to design systems which use CRTs with their larger screens. The problems of 20x22 character display are those of scale. The limited display size restricts the analyst in his design of system output formats. On the Picturephone it may take a bit more effort to produce useful output and may possibly require the division of related information onto several display pages but the data *can* be displayed and the executive *can* read and use it quickly. We feel that the problem of limited display size is more than offset by the fact that the Picturephone may be used both as a face-to-face communication device and as a remote terminal. Thus, its cost is essentially shared over both capabilities. In addition, the executive's desk is not cluttered with an additional screen and keyboard.

DATA BASE

For the initial presentation of APRIS to the top executives of Alcoa a sample of the type of information that could be efficiently and effectively displayed on the Picturephone was needed. The data had to be real and useful to the executives. It was felt that it would be a mistake to provide data that was either "dummied up" for the presentation, was of no use in the decision making process, or could be better presented by having it typed on a piece of paper.

There are two types of data which meet these criteria. The first is massive historical data which in hardcopy form is too bulky to allow convenient access. The second is data which changes more rapidly than can be routinely handled with current reporting methods.

For an example of the first, an existing consumer research data file was used. This file consisted of 40,000 data entries recording monthly shipment and production figures for aluminum and various consumer products ranging from vacuum cleaners to automobiles. The file was passed against a program which sum-

marized the data by year, quarter, and month and formatted it into approximately 4000 pages suitable for display. Another program created a series of index pages which allow any desired item to be located. An example of a typical inquiry, the yearly production of aluminum vans and the net change by year, with their associated graphs, is shown in the Appendix, Figure 5 through Figure 17.

As an example of the second type of data a daily report called the Forward Load Report was chosen. This consists of order information of various aluminum products produced by Alcoa plants. A program transforms the report into a displayable format and builds the indices necessary to access it. No example of this report is given here due to the confidential nature of the data.

CONCLUSIONS

The Picturephone has proven to be an effective and an efficient means of allowing executives to directly access a computer data base. However, Picturephone access, as an *isolated* capability, is of little use to a busy executive. Only when it is made one arm of an efficient information center does it serve to provide the executive with useful information for his decision making process.

APPENDIX

Example of a typical inquiry

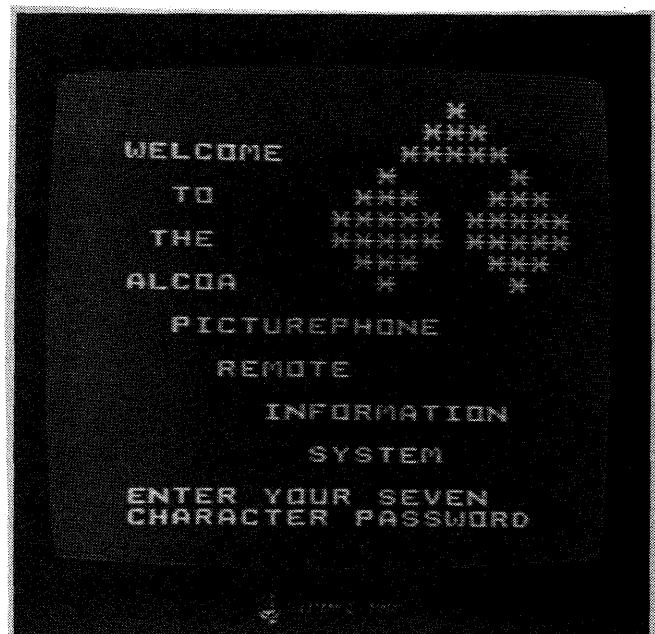


Figure 5—Welcome message

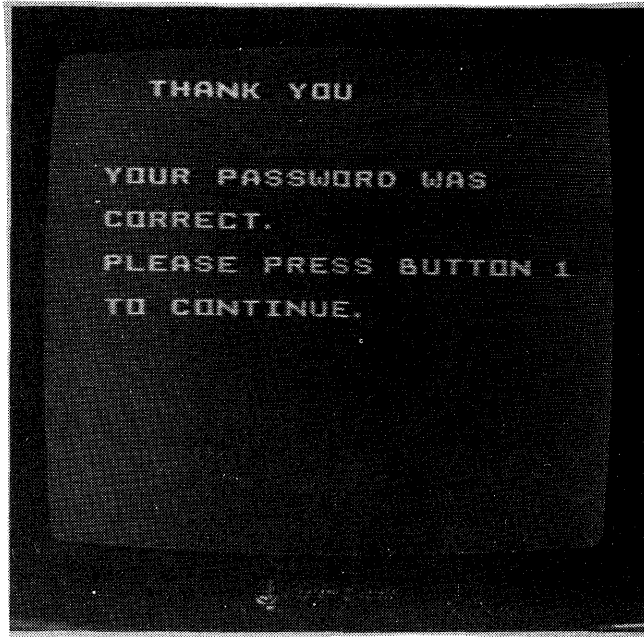


Figure 6—After entering password

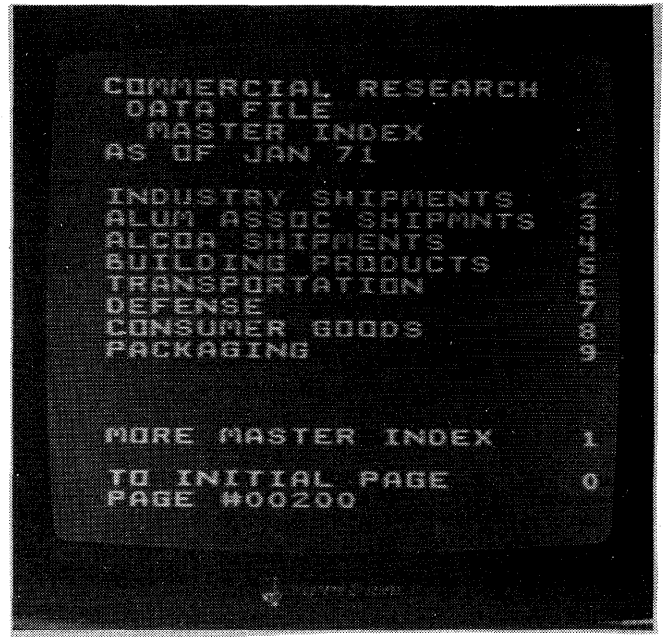


Figure 8—After pressing button 2



Figure 7—After pressing button 1

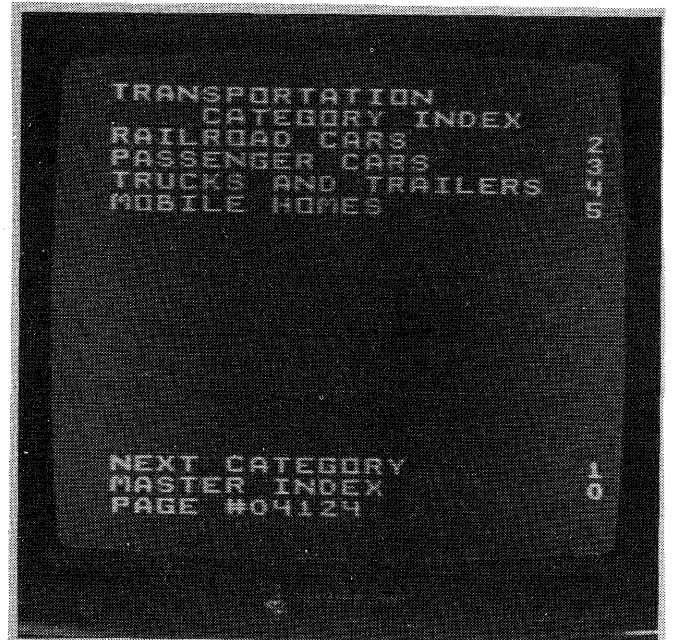


Figure 9—After pressing button 6

TRANSPORTATION TRUCKS AND TRAILERS	
TOTAL TRUCK PRODUCTI ON	2
GVW-6000 OR LESS	3
GVW-6001-10000	4
GVW-10001-14000	5
GVW-14001-16000	6
GVW-16001-19500	7
GVW19501-26000	8
GVW-26001-33000	9
MORE OF THIS INDEX TO CATEGORY INDEX	1 0
PAGE #04127	

Figure 10—After pressing button 4

ALUMINUM VANS	
DETAIL INDEX	
LAST 12 MONTHS	2
LAST 12 QUARTERS	3
LAST 12 YEARS	4
PRESENT MONTH, LAST 12 YEARS	5
PRESENT QUARTER, LAST 12 YEARS	6
NET CHANGE, 12 MNTHS	7
NET CHANGE, 12 QTRS	8
NET CHANGE, 12 YRS	9
BACK TO SUB-INDEX	0
NEXT DETAIL INDEX	1
PAGE #01902	

Figure 12—After pressing button 6

TRANSPORTATION TRUCKS AND TRAILERS (CONT)	
GVW-33000 & OVER	2
TRAILERS & CHASSIS	3
TOTAL VANS	4
STEEL VANS	5
ALUMINUM VANS	6
TOTAL TANK TRAILERS	7
DETACHABLE TRAILER BODIES	8
TO CATEGORY INDEX	1
BACK ON THIS INDEX	0
PAGE #04128	

Figure 11—After pressing button 1

ALUMINUM VANS	
LAST 12 YEARS	
1958	17120
1959	28791
1960	25475
1961	23560
1962	33421
1963	34478
1964	40823
1965	52365
1966	62884
1967	48758
1968	65248
1969	86714
PAGE #01905	

Figure 13—After pressing button 4

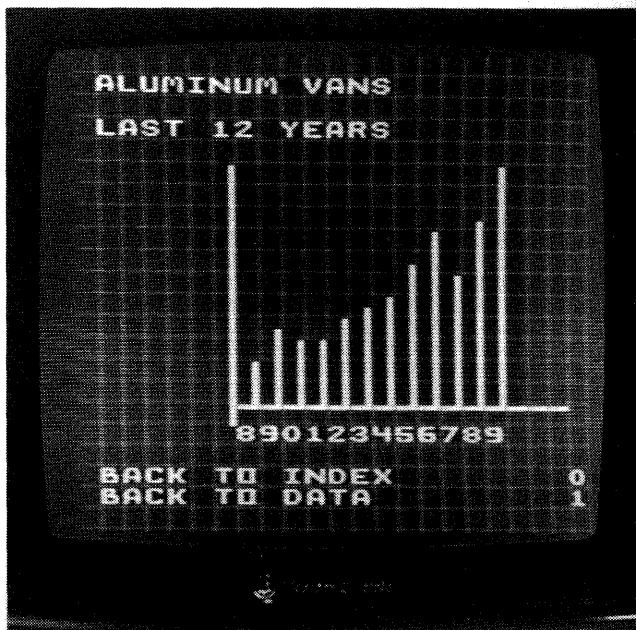


Figure 14—After pressing the code for graph: *1*

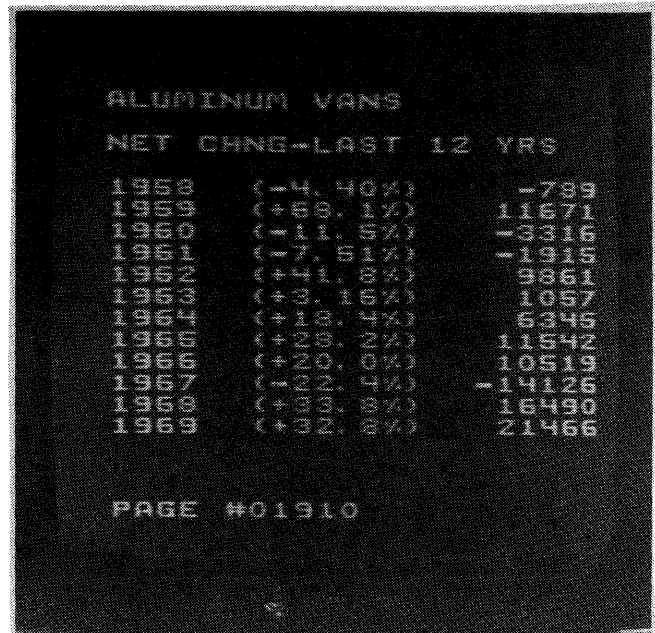


Figure 16—After pressing button 9

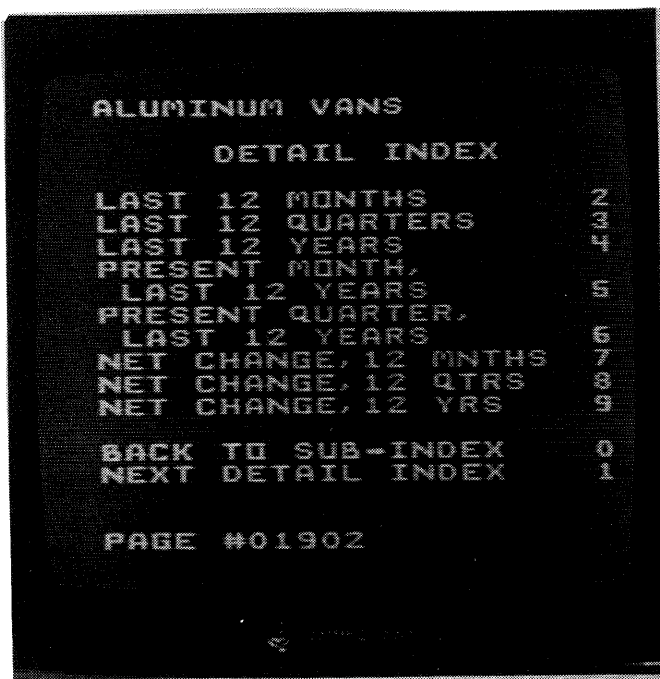


Figure 15—After pressing button 0

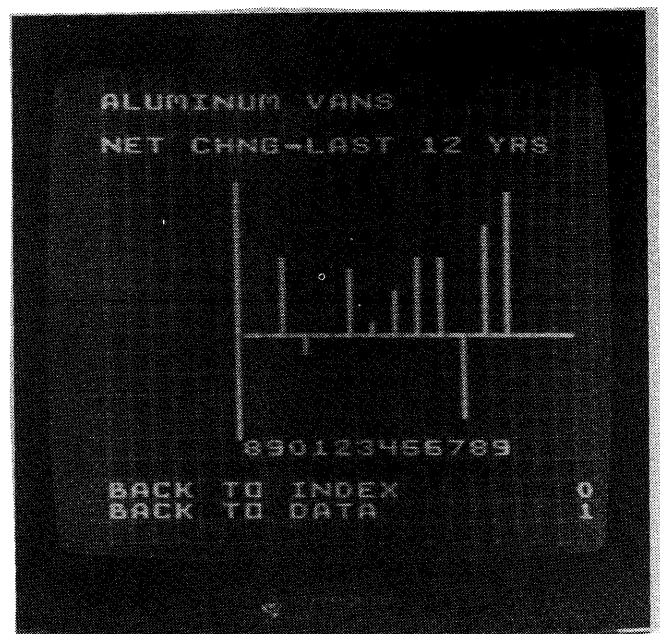


Figure 17—After pressing the code for graph: *1*

Computer support for an experimental PICTUREPHONE®/computer system at Bell Telephone Laboratories, Incorporated

by ERNESTO J. RODRIGUEZ

*Bell Telephone Laboratories, Incorporated
Holmdel, New Jersey*

INTRODUCTION

This paper describes the computer support of an experimental PICTUREPHONE/Computer system implemented at Bell Laboratories. The system provides Bell Laboratories and AT&T executives with the capability of using their PICTUREPHONE station sets to display information retrieved from a computer. Its primary purpose is to demonstrate the technical feasibility of accessing a computer from standard PICTUREPHONE stations and to help in the evaluation of the service.

Methods of operation in PICTUREPHONE/Computer systems can vary; they depend on the system objectives, types of users, and information to be retrieved. The information provided in this paper may serve as a general guide for those faced with the task of providing software and/or hardware to support PICTUREPHONE/Computer systems. The hardware and software used in the Bell Laboratories system are functions of the type of operation chosen and the particular *computer facilities which were available*. However, some of the concepts employed and techniques of overcoming implementation problems should be applicable to any PICTUREPHONE/Computer system and to a smaller degree, to the implementation of other systems which include terminals not supported by readily available hardware and software.

In the Bell Laboratories system, users gain access to the computer by dialing a PICTUREPHONE number associated with the computer. Thereafter, the user communicates with the computer using his station's TOUCH-TONE® dial. The computer's responses are displayed on the PICTUREPHONE station screen. A Display Data Set is used to interface the PICTUREPHONE network and station with the computer (see Figure 1). The Display Data Set translates TOUCH-

TONE signals into ASCII characters for the computer; it stores ASCII information from the computer and translates the information to an appropriate video signal for refreshing the display on the PICTUREPHONE screen. Thus, the computer is relieved of the task of repetitively transmitting the message to be viewed.

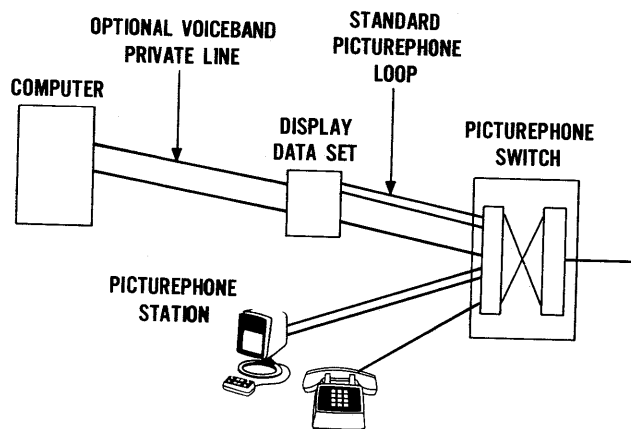


Figure 1—Computer access for PICTUREPHONE Service

Further information on the Display Data Set can be found in its Technical Reference* and in the February, 1971 issue of the Bell System Technical Journal. This paper is concerned with the software implementation and general operational characteristics of the Bell Laboratories experimental PICTUREPHONE/Computer system.

* Technical Reference Information for the Display Data Set Used to Provide Computer Access Service for PICTUREPHONE Stations—available from Engineering Director, Data Communications, American Telephone and Telegraph Company.

SYSTEM OBJECTIVES AND PROGRAMMING CONSTRAINTS

Man/computer dialogue in the Bell Laboratories experimental PICTUREPHONE/Computer system is constrained by certain physical considerations and human factors. The following physical considerations imposed general constraints on the system and consequently on the software implementation:

- The system is designed to be accessible from a standard PICTUREPHONE station set without requiring auxiliary input devices, such as keyboards, light pens, etc.; that is, only the TOUCH-TONE dial is required for input.
- Display size is limited to 440 characters, 22 characters per line, 20 lines per display.

Human factor considerations are particularly important when implementing this type of system. Basic assumptions in the design of the experimental system, which also imposed some constraints on the software implementation were:

- Users of the system would not, in general, have any knowledge of computer programming nor would they be willing to use a reference manual of interaction codes.
- User inputs should be as short as possible, without any intercharacter input time constraint and without the need for an "end of message" character.
- Each display should contain sufficient instructions to enable the user to select the next display, one of a few possible new displays, or return to a familiar point in the program.

In the experimental system, displays always contain enough instructions so that even the most inexperienced user may proceed from display to display with confidence. However, the more experienced users can use the system more efficiently, since more options are allowed at a particular point in the interaction than are enumerated on the screen. Once a user becomes familiar with the codes for selecting the system's various abilities, he often can choose them directly rather than being led step-by-step through a sequence of decisions. If a user makes a selection which is not allowed at a particular point in the interaction, he is presented an appropriate error display. This display tells the user the particular error he has made and gives him the option of returning to the point at which he made the error, reviewing a particular set of instructions, or using another code.

INPUT/OUTPUT HARDWARE AND SOFTWARE SUPPORT

General

The computer used in the Bell Laboratories experimental PICTUREPHONE/Computer system is an IBM 360/50 computer operated with the System/360 Operating System and under a multiprogrammed environment with a variable number of tasks (MVT). The computer is connected to the experimental PICTUREPHONE network via two Display Data Sets (DDSs). The DDSs are connected to the computer through an IBM 2701 Telecommunications Control Unit (hereafter referred to as the control unit) with two Terminal Adapters Type III. Since, in this system, the DDSs are remotely located from the computer, voiceband facilities equipped with 202D-series data sets are required on the transmission facility between the DDSs and the computer. Transmission is half-duplex at 1200 bauds.

Because the operational characteristics of the DDS are different than those of existing terminals and since the Bell Laboratories system represented the first use of PICTUREPHONE stations for computer access, it was not expected that standard computer hardware and software would support the system operation. Minor modifications were required in the input/output hardware and software to support the experimental system.

The terminal Adapter Type III (hereafter referred to as the adapter) normally permits the attachment to the computer of remotely located IBM 2260/2848 display complexes.

Operation with these devices involves polling and framing of messages, both of which require recognition of control characters by the adapter. However, the experimental PICTUREPHONE/Computer system uses simple Read only/Write only operations, without hardware recognition of line control characters, so that minor hardware and software modifications were necessary.

Hardware modifications

Two modifications to the adapter hardware were made to provide for the experimental PICTUREPHONE/Computer access system operation. These were to delete the two-second timeout period for the Read command and to delete the line-control character (EOT, STX, ETX, SOH, CAN, ACK, NAK) recognition. The first modification was made by grounding pin 01A-B2-C3-B09 and the second was made by grounding the "search latch" pin 01A-B2-G3-D06 in the adapter.

The two-second timeout was deleted in order to avoid the termination of the Read command if a character is not received within a two-second period. Although not an essential modification, the deletion of the two-second timeout would avoid the need to continually reestablish the Read command and thereby permit a more efficient computer operation.

The line-control character recognition was deleted because the associated polling mode of operation and associated message framing are not compatible with the PICTUREPHONE/Computer access system. Since only one PICTUREPHONE station is connected through a Display Data Set to the computer at one time, polling is not appropriate. Message framing would require the dedication of one of the 12 TOUCH-TONE input characters as an "end of message" character. This would restrict the user input capabilities and require users to remember to terminate inputs with a special character. This latter requirement was judged undesirable since the users are primarily members of upper management. The absence of message framing implies that the experimental software knows at all times how many characters it should expect and requires CPU intervention for each character received. Once a character is received, several validity checks are made and then a new Read command is issued; this process takes approximately 9 ms of CPU time and it repeats until the experimental software count is completed. Since the system was implemented a new "Read Clear" command has been made available which disables the "search latch" function.

Software modifications

The control unit operations are supported by two IBM data management access methods. One of these access methods, Basic Telecommunication Access Method (BTAM), which controls data transmission, was used to support the teleprocessing operations in the experimental PICTUREPHONE/Computer system.

BTAM is most helpful for implementing programs for telecommunications applications. It presently supports the following terminal devices: IBM 1030, 1050, 1060, 2260 and 2740 terminals, Bell System 83B3 and TWX stations and Western Union 115A stations. However, the use of BTAM to control transmission of computer messages to and from an unsupported terminal device not in the above list, such as the Display Data Set, requires special attention particularly when considering the use of the BTAM provided device input/output (I/O) modules.

A device I/O module contains the control information for the generation of channel programs for a given

terminal device. The terminal device supported by the adapter is represented by the device I/O module IGG019M3. However, since the experimental PICTUREPHONE/Computer system requires simple Read only/Write only operations, without line-control character recognition, and the module IGG019M3 did not provide for this type of operation, it was necessary for it to be expanded.

The following actions were taken to incorporate into the device I/O module the ability to support the experimental system:

- Two unassigned operation types representing Read and Write options in the 32-byte table of offsets were selected.
- Two entries in the channel program offsets for the operations were added. Each of these entries have a count of one for either a Read or Write operation and a pointer to a channel command word (CCW). A count greater than one is not necessary since there is no need for polling or acknowledgment of responses from the Display Data Set.
- Two channel command words for the Read and Write operations were added. They are:
01 04 00 00 20 11 04 00 for Write operations
02 04 00 00 20 11 04 00 for Read operations
In the above CCWs, the area address and count fields are obtained from the data event control block (DECB) associated with the Write or Read macro instruction.

PICTUREPHONE SOFTWARE

General

Software for the Bell Laboratories experimental PICTUREPHONE/Computer system operates in its own region of the computer. The PICTUREPHONE software handles several computer ports simultaneously, operates under an overlay structure, and consists of three self-contained but interrelated modules:

- Input/Output Telecommunications
- Executive Module
- Interactive Abilities

A region size of about 22,000 bytes is required by the Input/Output Telecommunications Module, which resides in core at all times and occupies the highest priority task. Because it occupies the highest priority task, jobs running in lower tasks are interrupted whenever the Input/Output Telecommunications Module

requires CPU attention. Whenever a call is received by the computer, a lower priority task is created by issuing an ATTACH macro instruction. This new task, having a maximum size of 18,000 bytes, contains the Executive Module and Interactive Abilities, which reside on disk.** When the call is completed, this task is removed from the system and its main storage area is released. The following sections will discuss some characteristics of the modules used in the experimental PICTUREPHONE/Computer system.

Input/output telecommunications module

This module is written in Basic Assembler Language (BAL) and in BTAM. It provides the program interface between the control unit, the Operating System, and the Executive Module for the PICTUREPHONE/Computer system. It controls the transmission and reception of messages between the computer and the PICTUREPHONE user by having the Operating System instruct the control unit to pass data to the Display Data Set and to receive data from the Display Data Set.

The Input/Output Telecommunications Module performs the following functions:

- Reads (i.e., Receives) characters from the Display Data Set via the control unit.
- Checks for the following ASCII control characters sent by the DDS:
 - ENQ—start of call
 - EOT—end of call
 - DC1—start of keyboard mode. In this mode, an optional adjunct alphanumeric keyboard can be used at the PICTUREPHONE station for input.
 - DC3—end of keyboard mode
 - SUB—start of edit mode. In this mode the user can modify the contents of the DDS buffer without interaction with the computer.
 - DC2—end of edit mode. This signals the computer that the user is finished modifying the DDS buffer contents. At this point the computer stores the modified display on disk or takes other appropriate action.
- Translates ASCII characters into numeric format.
- Stores numeric characters in common area accessed by Executive Module.

** If two simultaneous calls are placed to the computer, two new tasks of 18,000 bytes each are created.

TABLE I—Typical Abilities Provided in the Bell Laboratories PICTUREPHONE®/Computer System

1. AT&T Stock Report
2. Stock Market Report
3. Personnel Information
4. Bell System News
5. System Description (describes the system configuration and operation)
6. Calculator Routine (addition, subtraction, multiplication, division, square root functions are available)
7. Keyboard Routine (allows user to input information from an alphanumeric keyboard by means of the functions described in the Display Data Set Technical Reference)
8. Personal Files (files with personal information for a single user or a group of users)

- Creates new task by means of ATTACH macro instruction.
- Writes (i.e., Transmits) characters to Display Data Set via the control unit.
- Pending the receipt of an incoming input, or during the transmission of messages, causes the software package to enter a "wait state," i.e., it gives control to the Operating System to service programs in lower priority tasks.

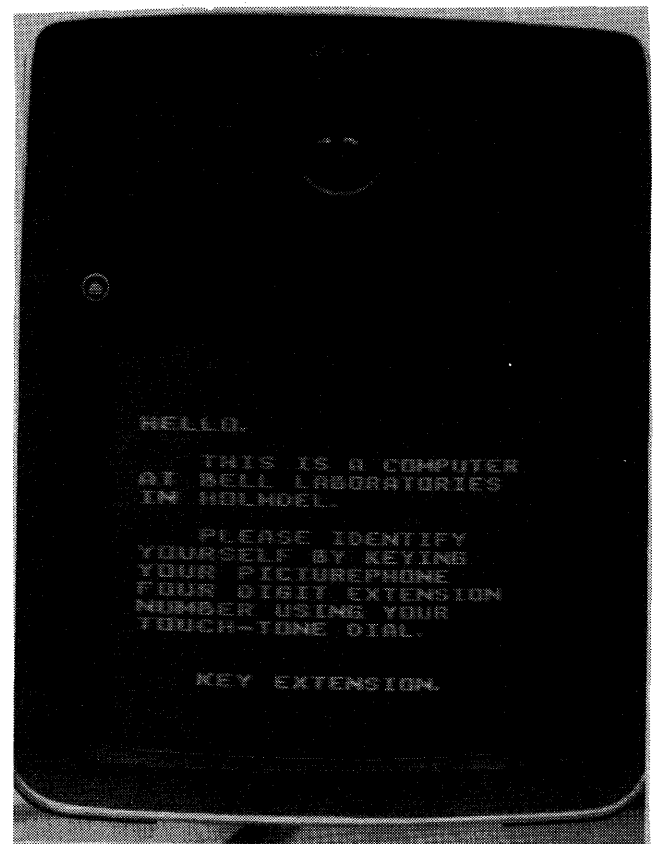


Figure 2—Hello message display

Executive module

The Executive Module is written in FORTRAN and it provides the logic for the selection of the abilities (see Table I). Basically, the Executive Module keeps an account of which display is being shown to the user and what course should be taken on the basis of his input. This module will handle the response itself when the response is a simple information retrieval application, e.g., AT&T Stock Report, or give control to one of the Interactive Abilities when the response requires additional processing, e.g., Calculator Ability. The Executive module scans every input from the user before handling any responses. The Executive's primary function for the user is the provision of general guidance to what information retrieval and other services the computer can provide.

Abilities

There are eight separate abilities (all written in FORTRAN) implemented in the experimental system.

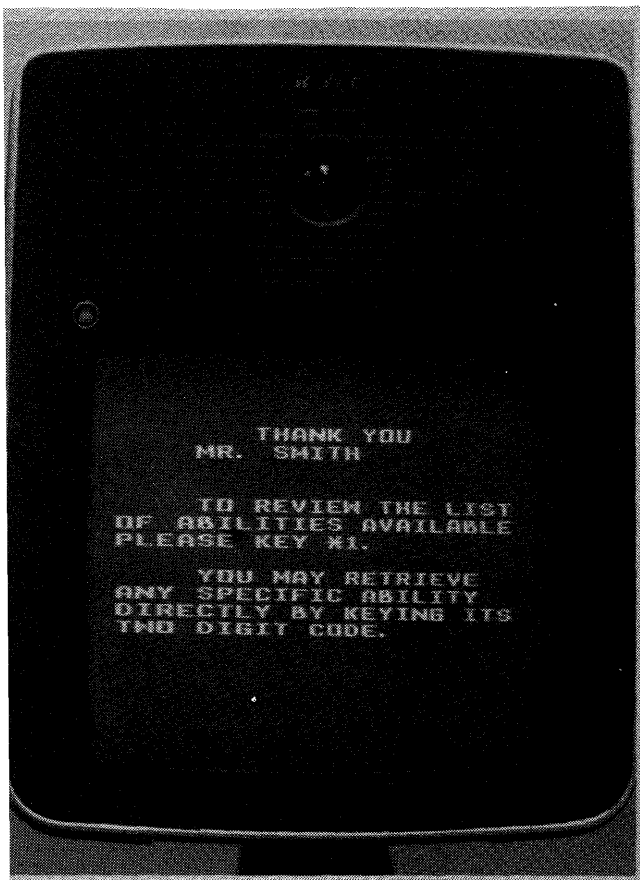


Figure 3—Thank you message display

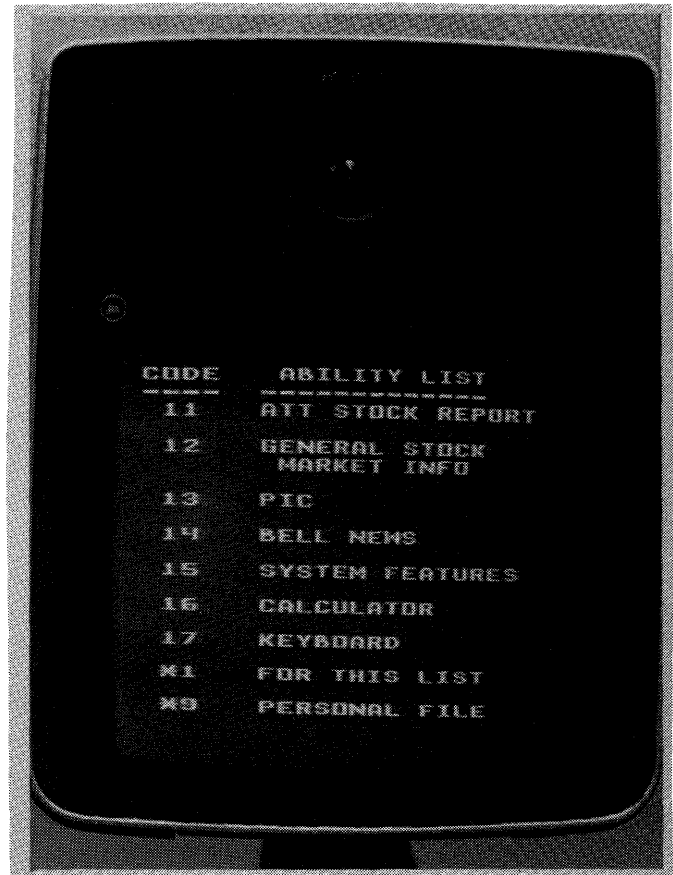


Figure 4—INDEX list

They are briefly described in Table I. An ability, when selected, will either handle subsequent dialogue itself until the user decides to leave it, or return control to a special section of the Executive which will interact with the user. This interaction is determined by certain choices, set by the ability, which are allowed to the user. Most of the abilities provide information retrieval functions and some of them require some degree of interaction. All information retrieval displays are stored on disk in fixed 440-character records. The abilities fit into a wide range of programming sophistication, from extremely simple to quite complicated. For example, the Calculator Ability allows the user to use his PICTUREPHONE station set as a desk calculator. Addition, subtraction, multiplication, division and square root operations, as well as memory, recall, start, and cancel features are provided. Obviously, the number of operations available is a function of the software design and is not limited by the DDS operational requirements. As in most of the other abilities, user-computer interaction is accomplished with the use of the TOUCH-TONE dial. A keyboard ability which allows the use

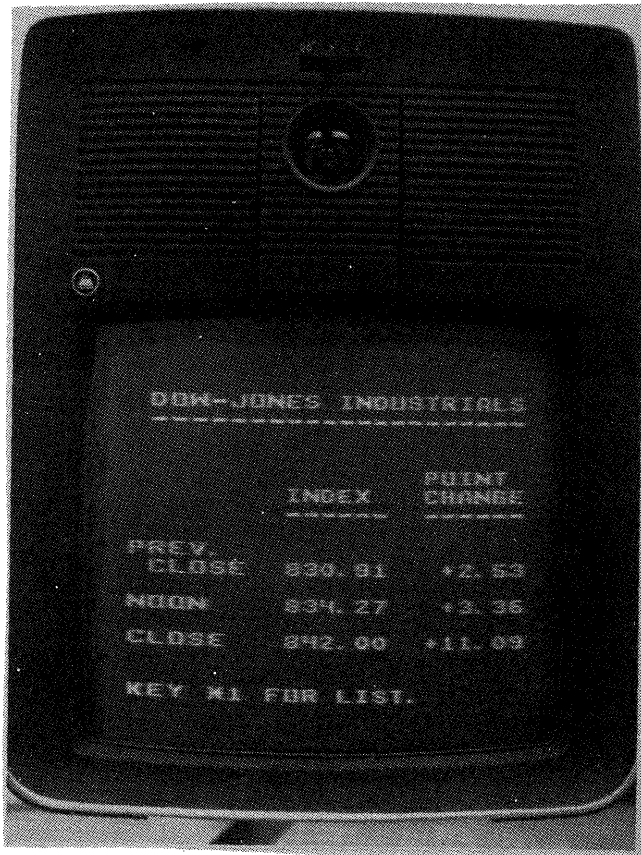


Figure 5—Stock market report display

of an adjunct alphanumeric keyboard to update information stored in the computer and which permits a greater input repertoire is also available in the experimental system. This ability allows the user to retrieve displays, modify them, and store the modified display on disk.

METHOD OF PROGRAM OPERATION

To use the experimental system, a user dials the PICTUREPHONE number associated with the computer and a connection is established. The Input/Output Telecommunications module recognizes the incoming call, gives control to the Executive Module, and provides a "Hello" message which is displayed with a request for the user's extension number (Figure 2). When the user inputs his four-digit extension number, the Executive Routine compares the extension number inputted with a list of valid user numbers. If a match is found, the name of the user associated with the extension number is displayed in a "Thank You" display (Figure 3). If an invalid extension number has been

inputted, the Executive Module will make the fact known and request the extension number again. After three illegal extension numbers, the user is instructed to hang up and get some help. At this point, the computer will not accept any more inputs from the user until a new call is made.

After a user gains access to the system successfully, he may request an INDEX of available abilities or he may go directly to any ability for which he knows the code. An INDEX display is simply a list of available abilities with their selection code numbers. From this INDEX any ability may be reached. The inexperienced user would naturally make use of these lists frequently. Figure 4 illustrates the INDEX list.

Most of the selection codes consist of two characters, with the INDEX pages and various abilities having permanently assigned codes. For example, the INDEX has code *1, Stock Market Report has code 12, etc. Figure 5 shows the format of the Stock Market Report display. Whenever the system is expecting a two character input, the user may select any ability, even in the middle of another ability. However, because of display size limitation, many of these choices

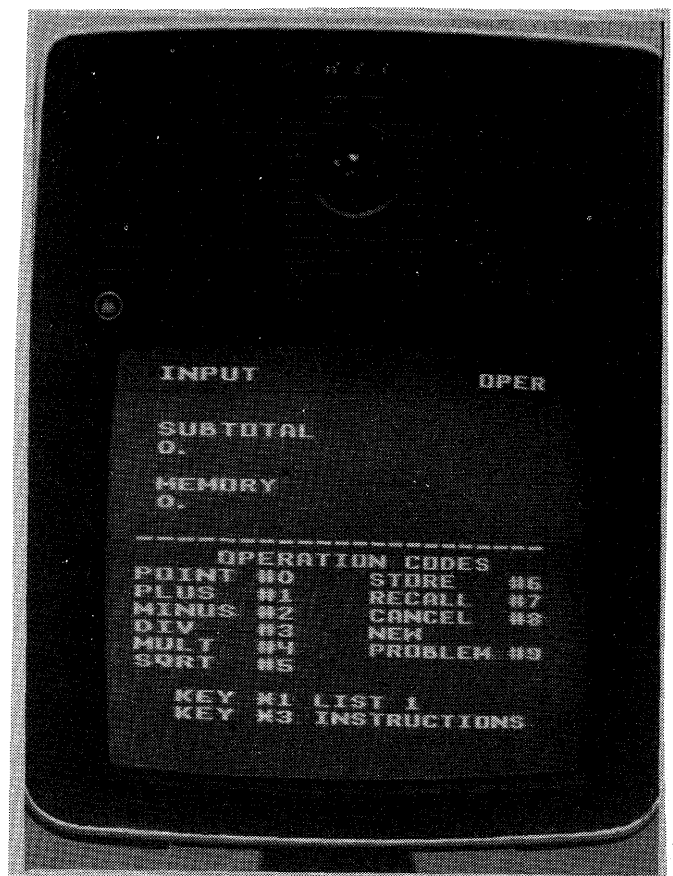


Figure 6—Operation codes in calculator ability

are not enumerated for him. Even in special routines, where an input of more than two characters is expected, the user may always opt for any of the *-plus-a-digit codes, which are legal at any time in the system. Thus, the user can always leave an ability at any point if he is somewhat familiar with the system. In any event, each display of an ability includes the code to return to the INDEX page. Within abilities, operation codes directed to the specific ability itself, are always 0-plus-a-digit. For example, an input of 01 may display the next page of a list.

Employing the various codes which the program displays, the user can command the computer's calculating ability. This is accomplished through the use of two-character operation codes. Figure 6 illustrates the set of TOUCH-TONE codes in the Calculator Ability.

CONCLUSION

General characteristics of an experimental PICTUREPHONE/Computer system at Bell Laboratories have been described.

The system has been operational on the Bell Laboratories corporate PICTUREPHONE network for about three years. It has resided in the same computer that serves other time-sharing applications, such as Con-

versational Programming System (CPS) and Administrative Terminal System (ATS). Although the experimental system is not at present part of a vital corporate information system, it is demonstrative of potentially useful PICTUREPHONE/Computer capabilities. In this system, the typical call holding time is about five minutes, the average number of retrieved displays is ten, and the approximate processing time is one second per call.

The method of operation described in this paper is simply one of several methods that could be used in implementing PICTUREPHONE/Computer systems. However, the information on the hardware and software used in the Bell Laboratories system may help those implementing systems in a similar environment. In addition, the techniques used to provide computer access to PICTUREPHONE terminals and the method of interaction employed at Bell Laboratories may be useful in designing PICTUREPHONE/Computer systems in different environments.

ACKNOWLEDGMENT

Mr. J. J. Mansell was initially responsible for the implementation of the experimental system. His guidance and technical advice are very much appreciated.

Proposed Braille computer terminal offers expanded world to the blind

by N. C. LOEBER

IBM Corporation
San Jose, California

INTRODUCTION

As professional people—engineers, scientists, programmers and managers—most of us make frequent use of the library. We keep abreast of recent developments by reading books or technical journals. If we have developed our reading skills, we can zip through a document at the rate of 1000 words per minute. But what would happen to us and our interests if we no longer had access to the library or if our supply of reading material was suddenly cut off because we became blind?

The end of the world you say! No, not quite, but it might seem so if this fate befell us. Unfortunately, more than 30,000 individuals lose their sight each year. The world hastily closes in on them. Books and magazines are practically out of reach. What alternative do they have to keep informed?

Thanks to Louis Braille and others, the Braille system of raised dots on paper provides an opportunity for written communication. Transcribing and embossing of Braille is difficult, so there is a limited amount of literary works available. This situation need not remain static. It can be improved by applying computers and programs to help translate Braille and to develop equipment for embossing Braille.

A brief tutorial on the raised-dot language of Braille is presented to illustrate some of its complexities. This is followed by an explanation of the methods used in producing Braille material. Finally, the proposed Braille computer terminal will be described and some experimental results from a feasibility model will be discussed.

THE BRAILLE SYSTEM

Braille was developed more than a century ago by Louis Braille, a French teacher of the blind, to provide

some means of communication for his students. It employs a system of embossed dots on the surface of the paper, which are felt and read with the fingertips.

Braille is read from left to right, top to bottom, exactly as a sighted person reads conventional printing. The average speed of reading is about 100 words per minute. Figure 1 shows the basic cell configuration for a Braille symbol. Up to six dots (two vertical columns of three dots each) are used. The dots of the cell are numbered as shown. Sixty-three dot patterns or Braille characters can be formed by arranging the dots in different positions and combinations. One other configuration, that of no dots, is used for spacing between words.



Figure 1—Braille cell dot identification

Figure 2 shows representative cell dimensions. The distance between the center of each dot is approximately $\frac{1}{10}$ inch. There are 4 cells per inch horizontally, and $2\frac{1}{2}$ lines per inch vertically. Dot height is about .020 inch.

Braille as officially approved in the United States includes several levels or grades, each level increasing in complexity with a corresponding reduction in the number of cells required. Grade I Braille provides full spelling of words and consists of the letters of the alphabet, punctuation, and a number of composition signs which are special to Braille. Figure 3 shows the basic Braille alphabet. Grade II Braille consists of Grade I plus 189 contractions in short form words and is officially known as English Braille. Grade II Braille is often compared to shorthand.

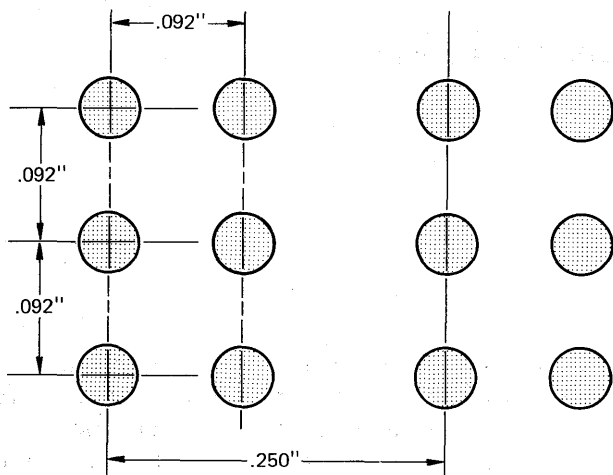


Figure 2—Braille cell dimensions

In between Grades I and II is another level of Braille which employs only 44 one-cell contractions. It is known as Grade I½. Grade III Braille is an extension of Grade II, using additional contractions and short form words and by the use of outlining (the omission of vowels). Grade III contains more than 500 contracted forms and is used mainly by individuals for their personal convenience. Several other Braille codes exist for special applications such as the writing of music and mathematics.

The majority of experienced blind readers use Grade II Braille. This is also used for most text printing because of the advantage of space saving (up to 30 percent), faster reading, and faster writing.

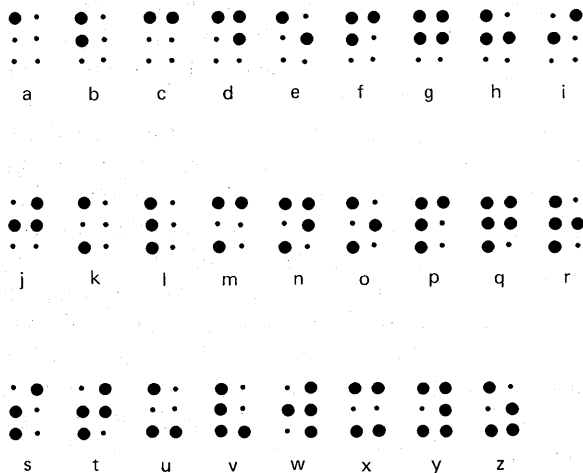


Figure 3—The English Braille alphabet

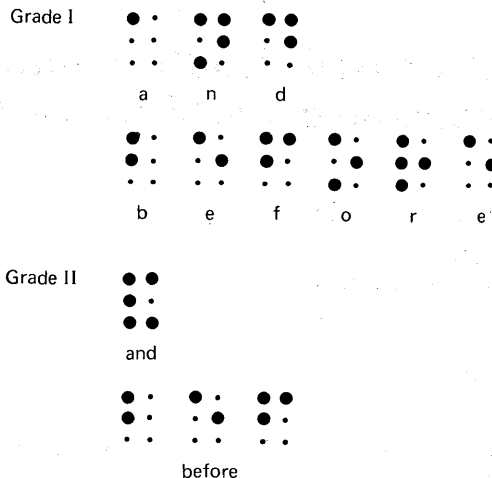


Figure 4—Word samples written in Braille

Grade I Braille utilizes a character-to-cell relationship. That is, each letter of a word would be reproduced as a Braille cell. This is slow reading and results in a bulky transcript. However, it is necessary to use this level of Braille for writing programs and statistics. As a point of information, there are now approximately 400 individuals who have been trained as programmers. Although handicapped, these programmers are active and productive workers in our society.

Conventional spelling is perfectly feasible in Braille and is used in some applications such as computer programming, but the more frequently used contractions are assigned their own dot configurations. Some cell combinations are used either as a whole word or part of a word or possible letter groups. In English Braille, for example, the letter "f" when alone (or adjacent to a punctuation mark, the capital sign or the italic indicator) stands for the word "from." The "ch" sign under these same conditions means "child." This method of writing is known as contracted Braille, and the characters used in this way are called Braille contractions. The method differs from regular shorthand in that by assigning actual letter group values to most

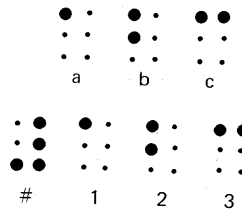


Figure 5—Use of number sign

of the contractions, conventional spelling is significantly retained in spite of the contractions.

Figure 4 gives a comparison of Grade I and Grade II Braille. Because of the many contractions used and the particular rules that apply to the hyphenation of words, the capitalizing of letters, and the displaying of numbers, it is best to consider Braille as a foreign language. It requires training, skill and practice to be a good transcriber or to write and read Braille.

Numbers are represented by using the first 10 letters of the alphabet. Figure 5 shows the "number sign" preceding the letters, the scheme for converting them to numbers. Figure 6 shows how the "capital sign" is used. A single capital sign tells the reader that the following letter is capitalized. Two consecutive capital signs indicate the entire word is capitalized. Thus, we see that interpretation of Braille is based on adjacent cells as well as the cell being read.

BRaille PRODUCTION

Braille documents have been produced by using a variety of devices. Some of these are reviewed here.

Braille slates

Figure 7 shows a typical Braille slate, guide and stylus. Individual cells or dot patterns are manually and singly embossed with a stylus which is guided across the writing line by a metal strip or guide. Braille embossed on a slate must be the mirrored image of the actual embossing desired, because the dots are formed on the back side of the paper. Reading the dots necessitates reversal of the paper. This is essential since the depressions cannot be felt. To make a correction, the paper is removed from the slate, and the dots are flattened with a blunt instrument or correcting tool.

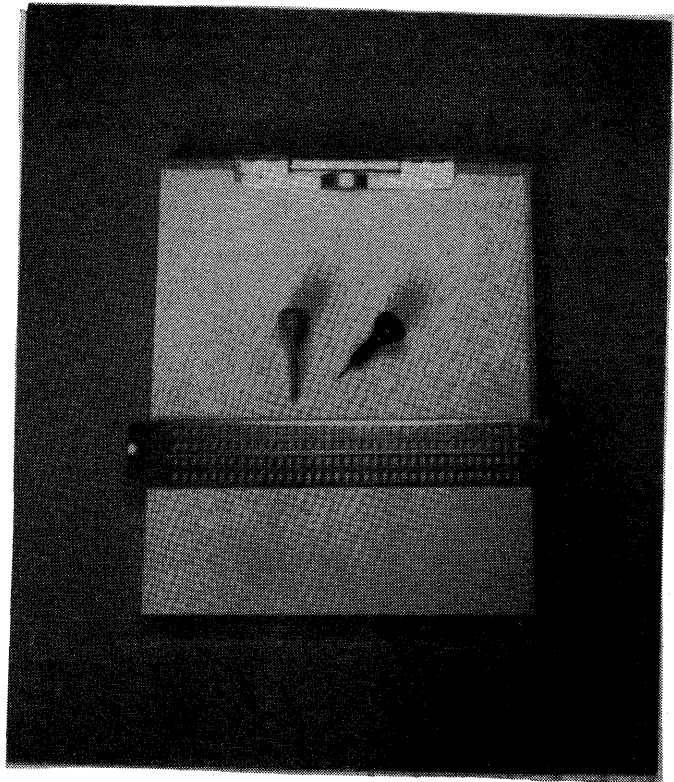


Figure 7—Braille slate

Braillewriters

These are similar to small portable typewriters. Figure 8 shows a Perkins Braillewriter made by the Howe Press Company. There is a key for each of the six possible dots. From 1 to 6 keys must be depressed

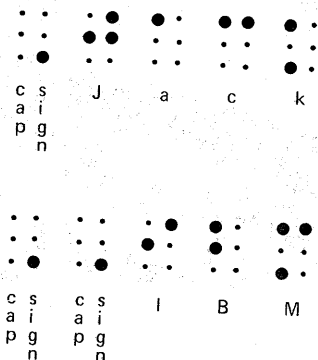


Figure 6—Use of capital sign



Figure 8—Perkins Braillewriter



Figure 9—IBM Braille electric typewriter

simultaneously to emboss a Braille cell. The embossing usually occurs from the back of the paper so that the normal left-to-right reading is possible and the transcriber may check his work as he goes along.

Braille typewriter

Figure 9 shows an IBM Electric Braille Typewriter with a full alphabetic keyboard. The usual type faces have been replaced with dot configurations. Embossing is from the front into a rubber platen. This typewriter is easy to use because only one key is depressed at a time for any particular cell combination rather than multiple keys as on the Braillewriter.

Press braille

Where large quantities of the same material are needed, metal master plates are made on a stereotype machine such as the one shown in Figure 10, built by American Printing House for the Blind (APH). These master plates are then used on various types of presses for embossing Braille. An example of this might be the Braille edition of various magazines, periodicals, or religious books. Figure 11 shows a typical book of interpointed press Braille.

Unfortunately, due to the wide variety of textbooks used not only throughout the United States but even within one state or within a school district, it is not

feasible to produce textbooks in "press" Braille. Most textbooks, school materials and the like are produced by volunteers utilizing hand or manual embossing devices. There are several Braille printing houses which attempt to fulfill the need of general-interest Braille material. The cost of producing this Braille is partially defrayed by Government agencies. Some books, magazines, and other publications are generally available from the Library of Congress.

Computer braille

Some progress has been made in the production of Braille on high-speed printers coupled to computers. Generally, a single copy is produced and has limited life, depending on the paper used and the method of embossing. (Properly embossed Braille on special paper usually lasts for 50 readings.) Embossing by impacting against a rubber platen does not produce as well a defined dot as when a metal die is used. The rubber platen tends to mushroom the dot base and limit dot height.

During the past few years, several organizations have written programs for various computers. Some of these are used regularly to produce Braille output for the blind. Printouts may be computer translated Braille from English input or conversion of computer output to simple Braille as might be used by a blind programmer.

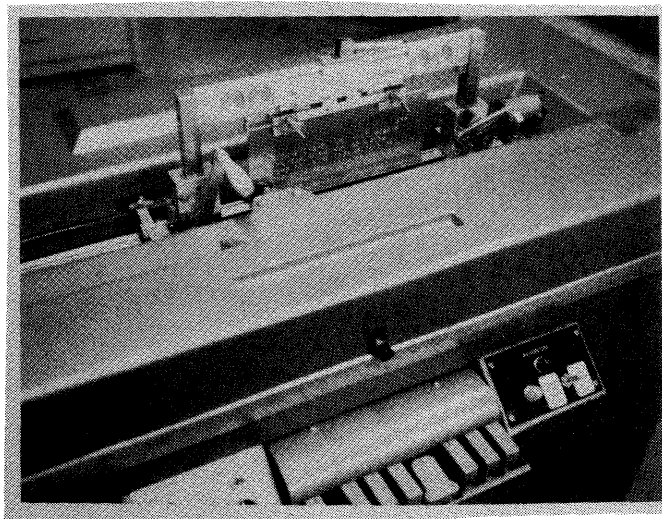


Figure 10—A Braille stereotype machine built by American Printing House for the Blind

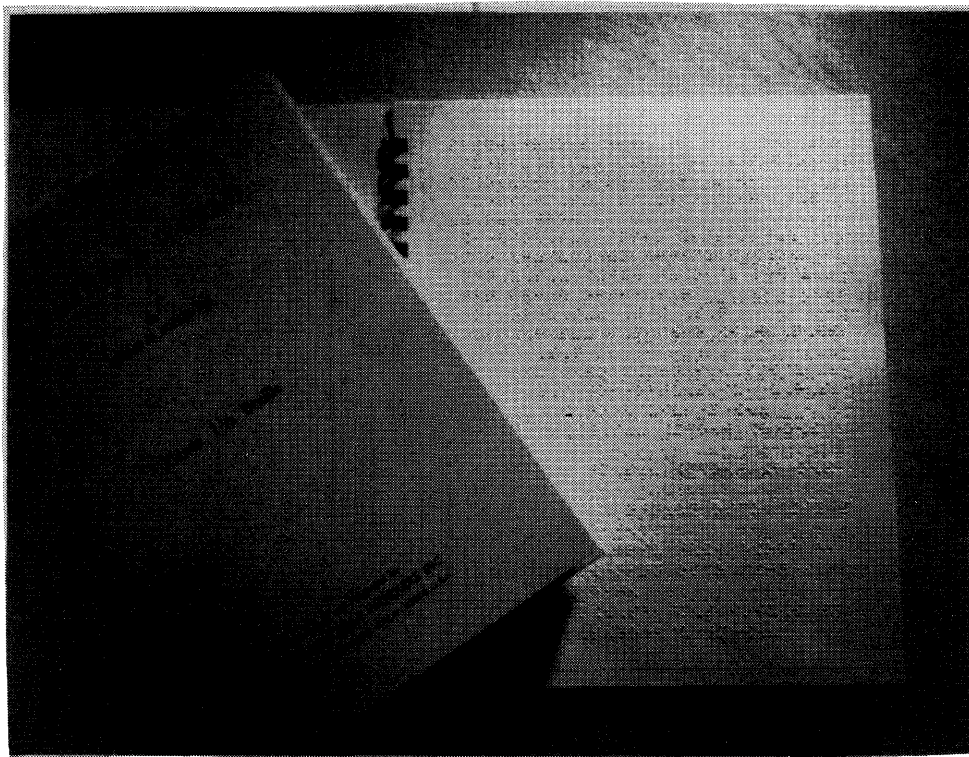


Figure 11—Braille book

PROPOSED ON-LINE BRAILLE TERMINAL SYSTEM

Description

Figure 12 shows a typical on-line terminal. Such terminals are attached by communications lines to a system to provide real-time response to various inquiries and computer assistance with mathematical problems. This capability is available now. It's a matter of using existing technologies to develop a system that will greatly improve communications and facilitate the availability of information in Braille.

The key to the proposed on-line Braille terminal system is the embossing terminal printer. This system, supported by some programming, would open "Pandora's Box" to the blind.

The terminal system should be versatile enough to operate in two modes, first as a local typewriter unit and second on-line to a computer. In the local mode the input terminal keyboard would be modified to provide the Braille function keys such as the number sign, capital sign, etc. A possible keyboard layout is shown in Figure 13. This keyboard includes all the Braille function keys, as well as the English Braille

contractions. The configuration shown is used by the Lutheran Braille Workers, Inc., on their modified IBM keypunches. These automatic Braille transcribing keyboards have been used since 1956 and have proven very satisfactory.

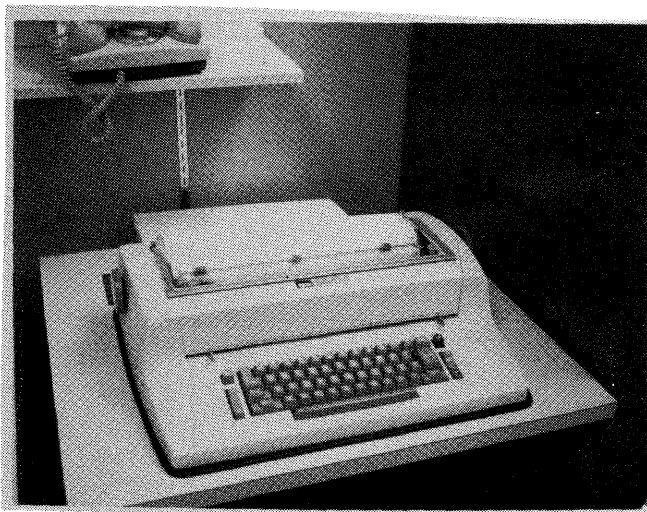


Figure 12—On-line terminal

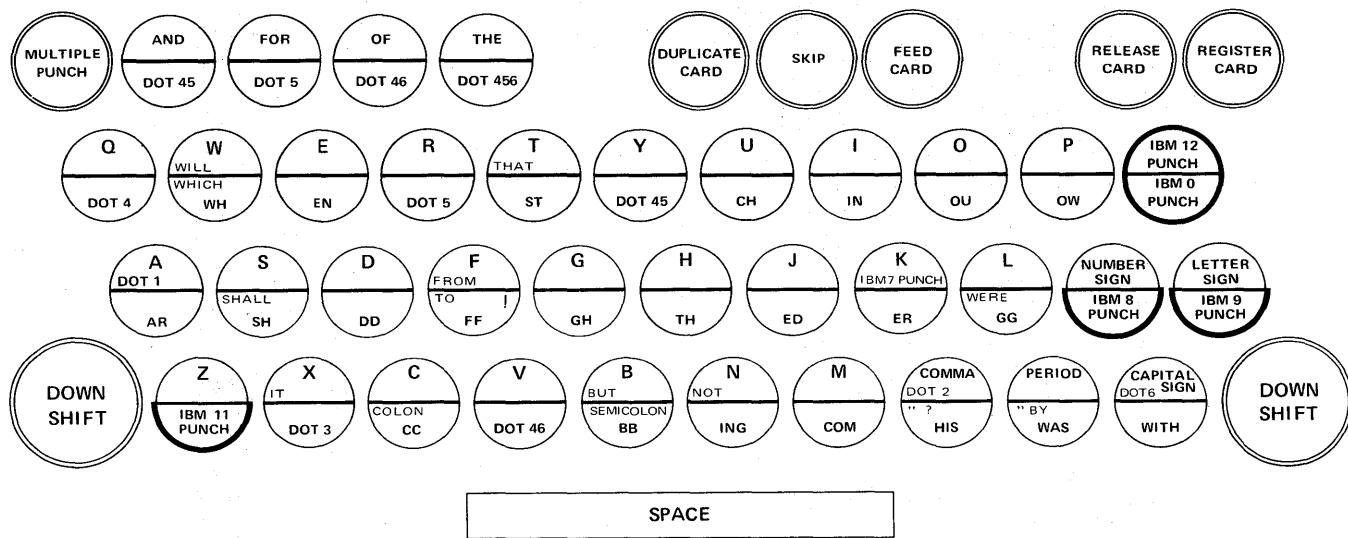


Figure 13—Automatic Braille keyboard as used on IBM keypunch

Figure 14 shows a diagram for a possible terminal system in the local mode, with the ink-print in/out terminal, encoder and a second unit for embossing Braille. Ink-print copy can be made available for the sighted and embossed copy for the blind. The embossing mechanism, having been carefully designed from a human factors standpoint, allows reading of the dots immediately after embossing without need of moving the paper. This is especially helpful to the blind person if he is interrupted while typing. It means that he can

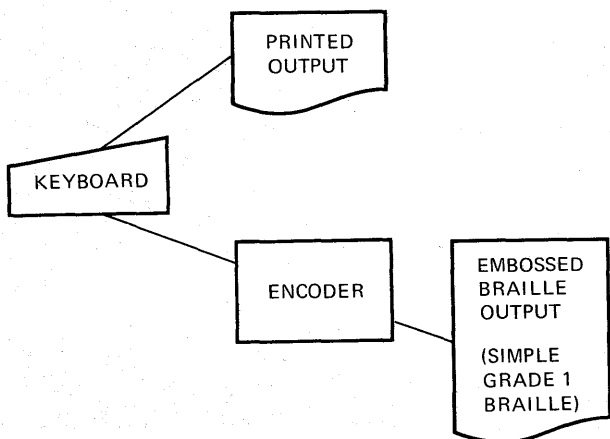


Figure 14—Proposed Braille terminal system with embosser (local mode)

read the embossed copy to review what he has written. A metal die is used to ensure a good quality dot.

Figure 15 shows the on-line operation. When the embossing terminal is used on-line to a computer,

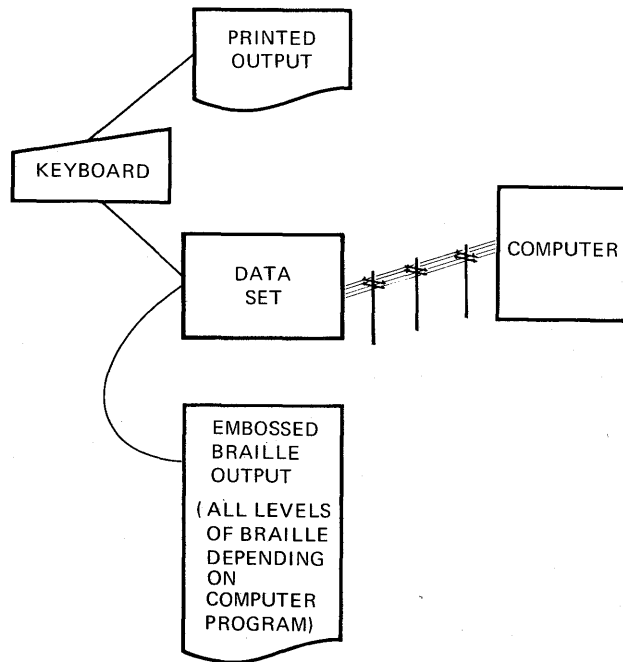


Figure 15—Proposed Braille terminal system with embosser (on-line mode)

various conversion or translating programs resident in the computer would assist the individual. These programs would convert information in various reference banks to the Braille codes.

Examples of possible system operation

Many sighted programmers use on-line terminals to write their programs and communicate directly with the computer. This speeds up the process of writing and debugging programs. The sightless programmer does not have this advantage. He does not have two-way communication with the computer. He must depend on Braille output from a high-speed printer which is usually run only once a day because certain modifications and special setups are required to print Braille. If we could provide the sightless programmer with his own embossing terminal, it would greatly increase his communication ability. To bring this about, it is necessary to take the programs that are used to provide Braille output on a high-speed printer and modify them for use on a Braille printout terminal. Providing this capability would mean many additional job opportunities for the blind.

Let us consider another example. Many school districts are now training sightless children in the classroom along with the sighted. This is an excellent arrangement in that it does not separate the handicapped child, but rather places him in society where he can participate and learn to get along with others. However, it is not an easy arrangement for the teachers, the school, or the students. Text material, handouts, and test papers must be provided in Braille for the child. The production of these various documents can at times be very awkward, inconvenient, and almost impossible.

At present it is necessary for the teacher either to know Braille and transcribe and emboss a document into Braille for that student or to enlist the aid of a volunteer to do this for him. Often there is a lack of time or availability of a trained transcriber to do this. A school district utilizing a central information bank could have much of this information available on tapes or on-line storage. When the teacher needed a copy of a particular examination, he could request it by phone and it would be provided to him, perhaps hundreds of miles away, by means of the on-line embossing terminal.

In cases where a document is not on file, the teacher could enter the text by typing it into the computer. The computer would then translate this and provide the embossed Braille on a real-time quick turnaround basis. If a terminal was used to prepare the ink-print master copy of the test, it could also be transmitted to the

computer for translating. The Braille copy would then be available simultaneously with the master copy of the test for the sighted individuals. Such an arrangement would greatly aid the educational process and give the handicapped child many of the advantages and opportunities provided to the sighted.

A third example is the blind child who is limited by the number of reference books that are available for him to do his homework. He really can't afford to own a personal copy of some books. Besides being expensive, the books are voluminous, requiring much storage space. For example, a 30-volume encyclopedia used by a sighted person would equal 145 volumes of 4- to 5-inch books in Braille.

The use of a remote terminal and various data banks or information systems could solve the problem nicely. It is not hard to realize or project that a blind student could have a terminal in his home and proceed to do his homework by dialing into a data bank and inquiring about the particular subject of interest to him. Imagine the benefits to this individual if he could dial into a dictionary or an encyclopedia. What a tremendous boon to him to be able to inquire on a particular subject and have the computer respond by embossing on his remote terminal the information he is seeking.

Projecting even further we can see where a low-cost embossing terminal could be installed in the home of a sightless person for daily communication. Major newspaper items and magazine articles could all be made available from the central information bank. Individuals could receive these by means of their telephone and the Braille terminal. They could have access to many of the same articles that you and I are privileged to read.

Our last example covers handicapped individuals who have other problems in addition to blindness. They too could benefit by having a terminal in the home. Specifically, this arrangement could provide an opportunity for a new productive life. The individual, although afflicted with immobility or other difficulties, could work in his home and contribute to society. He could be employed as a programmer, communicating with the computer, developing programs, receiving his response from the computer, and enjoying two-way communication.

RESULTS OF EXPERIMENTAL MODEL

An experimental model of the proposed system was built and used for various feasibility tests. Figure 16 shows the model which consists of an ink-print terminal with an attached Braille embossing unit. Special attention was given to the human factors requirements

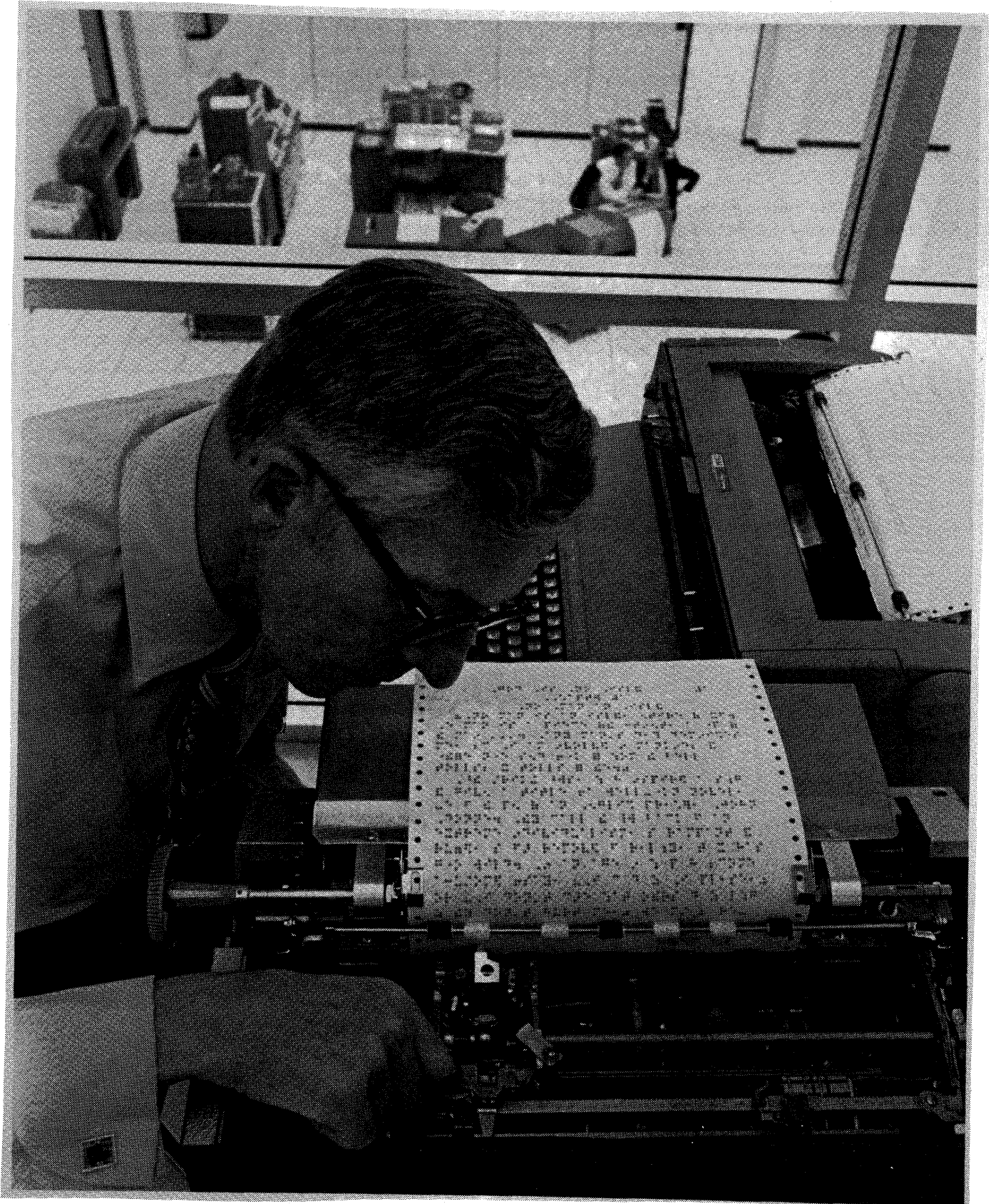


Figure 16—Feasibility model of on-line terminal with Braille embosser

during the design stage. On the basis of this study, the unit was designed and built to emboss from the rear, with the data appearing on the front side of the paper. A metal die was used to mate with the selected pins to provide positive control in forming the raised dots.

Results of the feasibility tests have been favorable. Quality of the Braille dots is good, making the symbols easy to read. Front embossing offers added convenience to the blind operator in that fingertip reading and checking are possible while the paper is in the terminal. Braille printout for the blind and conventional printout for the sighted, both from the same terminal system, allow improved speed in communicating between each other.

Our investigation and experimentation with the Braille terminal will continue. We hope to define a practical, easy-to-use, general-purpose embossing terminal. To accomplish this, we are continuing to discuss and

explore the actual use of such a terminal with additional blind individuals.

BIBLIOGRAPHY

- 1 R B STEWART JR
Suggestions for curriculum and ancillary services in training the blind to program computers
System Development Corporation Publication
No. PB-176-841 1968
- 2 N C LOEBER
Automatic Braille keyboard
IBM Corporation San Jose California Publication
No TM 02 138 1960
- 3 AMERICAN FOUNDATION FOR THE BLIND
Understanding Braille
New York New York
- 4 AMERICAN PRINTING HOUSE FOR THE BLIND
General catalog of Braille publications
Louisville Kentucky 1969

Numerical simulation of subsurface environment

by BARRY L. BATEMAN

University of Southwestern Louisiana
Lafayette, Louisiana

and

PAUL B. CRAWFORD and DAN D. DREW

Texas A&M University
College Station, Texas

INTRODUCTION

Subsurface environments are heterogeneous. The permeability and porosity varies from point to point and permeability is only approximately related to porosity. To construct a numerical model that is useful in the analysis of the flow of oil, gas or water in the stratum one must simulate the porous media.

When permeability and porosity values have been assigned for each sector of the strata, they may be used to calculate the fluid saturation at geological equilibrium.

In this paper emphasis has been placed on simulating realistic rock properties throughout the reservoir.

Advances in reservoir engineering were being accomplished at a rate comparable to those in numerical techniques. Most early work assumed a homogeneous rock matrix. It was generally accepted that the reservoir had one permeability, porosity, initial water saturation, and one capillary pressure curve that was constant through the reservoir.^{1,2} The error of this concept was well-known, but solutions to more realistic approaches were too difficult. Instead of a completely homogeneous reservoir, heterogeneous reservoirs consisting of two homogeneous layers have been studied.^{3,4} This corresponds to stratified or layered systems. Porous rock that is normally considered homogeneous will have small to large variations in its porosity and permeability when sampled at different areas. Although these variations exist, these properties will have a maximum, minimum and an average value. Analysis of field data indicates that these values are not predictable, but can be represented by a distribution about some mean value.

One of the basic prerequisites in performing a simulation is the availability of easily attainable random numbers. In this study, uniformly distributed random numbers between zero and one were generated by a modified version of IBM's RANDU routine.⁵ Several methods are available for generating various statistical distributions from uniform random numbers.⁶ The use of the cumulative distribution form of the desired distribution is among these methods. This method utilizes the fact that the range of both the uniform distribution and the cumulative function is between zero and one. Using this fact and the limiting parameters on the second distribution, it is convenient to randomly generate numbers from the desired distribution.

PERMEABILITY

Measurements of permeability are given in darcys. A rock of one darcy permeability is one in which a fluid of one centipoise viscosity will move at a rate of one cubic centimeter per second under a pressure gradient of one atmosphere per centimeter and a cross-section of one square centimeter.⁷ Since this is a fairly large unit for most producing rocks, permeabilities are commonly expressed in units one thousandth as large, the millidarcy.

The permeability of an oil-bearing and commercially productive sandstone formation is normally between ten and five hundred millidarcys. The actual permeability range, as well as the distribution of permeability values, is determined from core data of the formation to be simulated. The distribution of permeability values is ordinarily not uniform. In a typical rock formation

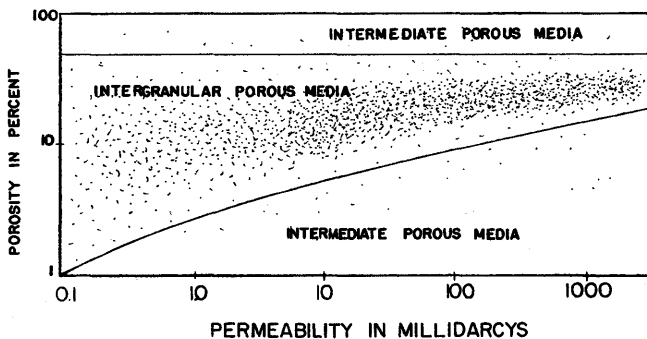


Figure 1—Porosities and per meabilities of 2200 sandstone specimens

there will be a number of values within the range that will be points of concentration for the measured values. These values are called cluster points. The cluster points need not be well defined and the distribution of values around the points can be irregular, but this concept is quite useful in representation of geological strata. Figure 1 is an illustration of this distribution.⁸ It gives a plot of the porosity versus permeability values of 2200 sandstone specimens of two types: intermediate porous media and intergranular porous media. We are concerned with the area between the two curves which represents the intergranular porous media. Considering only the permeability values and not the porosity values, one can detect several poorly defined cluster points.

This clustering effect can be generated using the following procedure:

1. Assume a permeability range X_0 to X_n .
2. Choose values of X_i such that $X_0 < X_1 < X_2 < \dots < X_{n-1} < X_n$.
3. Assign percentage values, k_i , to each of the segments, X_{i-1} to X_i , of the permeability range. These percentages determine the number of permeability values which will be generated for each of the segments. This allows the clustering effect to be simulated.
4. Calculate

$$A_i = \sum_{j=1}^i k_j \quad i = 1, 2, \dots, n$$

where $A_0 = 0$.

5. Generate a random number Z such that

$$0 < Z < 100.$$

6. Find m such that $A_{m-1} < Z < A_m$
7. Set the permeability value equal to

$$(A_{m-z}) / (A_m - A_{m-1}) (X_m - X_{m-1}) + X_{m-1}$$

After the calculations for the first four steps have been completed, steps 5, 6, and 7 are performed repeatedly until a permeability value has been calculated for each computational module. The sequence in which these values are assigned to the modules is not important.

The 1225 permeability values were generated in this manner. The permeability range was 10 to 500 millidarcys. In this instance $n = 10$ and

$X_0 = 10$	
$X_1 = 59$	$k_1 = 15$ percent
$X_2 = 108$	$k_2 = 10$
$X_3 = 157$	$k_3 = 10$
$X_4 = 206$	$k_4 = 10$
$X_5 = 255$	$k_5 = 10$
$X_6 = 304$	$k_6 = 10$
$X_7 = 353$	$k_7 = 10$
$X_8 = 402$	$k_8 = 10$
$X_9 = 451$	$k_9 = 10$
$X_{10} = 500$	$k_{10} = 5$

It can be seen that more permeability values are found in the 10 to 59 millidarcy section of the range than in the 451 to 500 millidarcy section of the range. The other values are relatively uniformly distributed.

The permeability values shown in Figure 2 were assigned to the modules of the eight by eight grid system illustrated in Figure 3. The shaded area of each module is proportional to the permeability value assigned. For example, the permeability value of the upper left-hand module is 105 millidarcys.

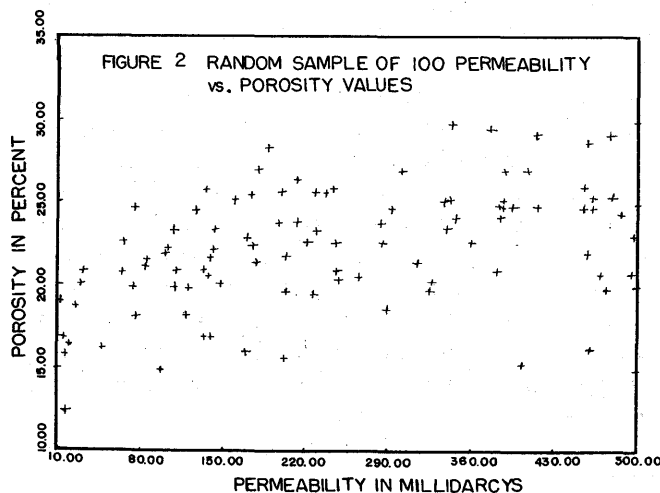


Figure 2—Random sample of 100 permeability vs. porosity values

POROSITY

From the reservoir engineering standpoint, one of the most important rock properties is porosity, a measure of the space available for storage of water and oil. Porosity is defined as the ratio of the void space in a rock to the bulk volume of that rock expressed in percent.⁹ The porosities of oil-bearing and commercially productive sandstone formations generally lie between ten and thirty-five percent.⁸

The nature of the relationship between permeability and porosity is illustrated in Figure 1. The porosity and permeability in 2200 specimens of sandstone was measured and this plot was made of porosity versus permeability. It can be seen that there is an approximate linear relationship between the two properties; however, for a given permeability value there is a corresponding range of porosity values. The porosity values within this range are normally distributed about a mean value with a small variance. Porosity values having this relationship to permeability, as shown by Muskat,⁸ can be calculated readily from a function of the form:

$$\text{Porosity} = f(A, B, \text{Sigma}, \text{Permeability})$$

where A and B are coefficients of the linear relation between the average value of porosity and the permeability value, and Sigma is the prescribed standard deviation.

The porosity values illustrated in Figure 4 for an 8×8 grid system were assigned using the function described previously. Like Figure 3, the shaded area in each block is proportional to the porosity value assigned to the module. For example, the module in the upper

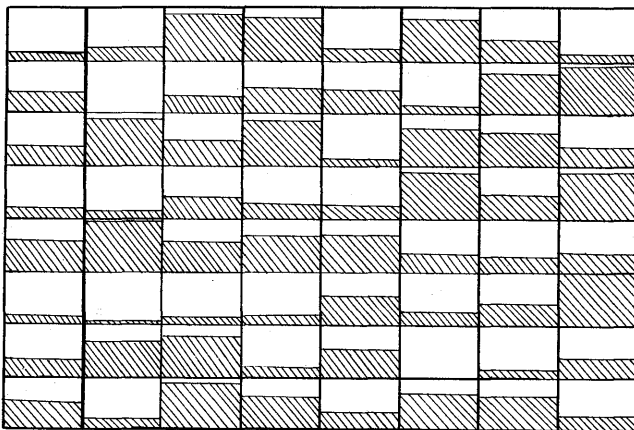


Figure 3—Typical reservoir permeability by blocks
(10 to 500 md)

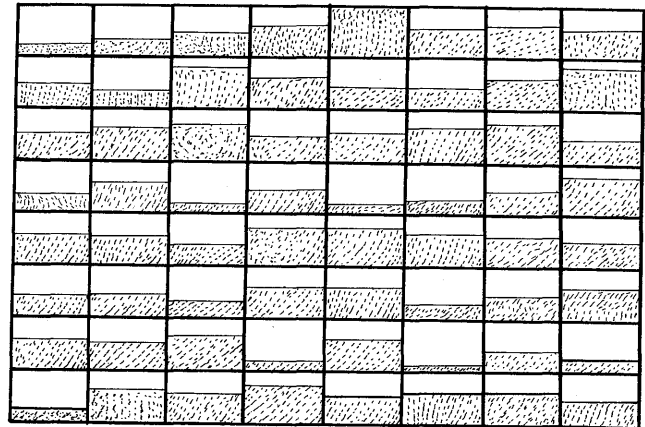


Figure 4—Typical reservoir porosity by blocks (10% to 35%)

left corner of Figure 4 represents a porosity value of fifteen percent.

The algorithms for porosity and permeability values were programmed for a digital computer. This program was used to generate 1225 points. The distribution of these points reasonably duplicates the actual sample values presented in Figure 1. Having A, B, and Sigma as input parameters to the algorithms, as well as the ranges for permeability and porosity, enables one to easily duplicate results obtained from actual measurements of samples for a given formation. It should be noted here that particular measurements made in a rock formation are not duplicated. The method duplicates the general rock properties of the strata.

CAPILLARY PRESSURE

Because oil-bearing reservoirs universally contain more than one fluid phase, interfacial forces and pressures are continually influencing both static and dynamical states of equilibrium.⁸ The pressure difference across an interface between two fluid phases is the capillary pressure in dynes per square centimeter. When this is expressed in oil-field terms, the equilibrium capillary pressure in pounds per square inch can be stated as:

$$P_c = h/144(\rho_1 - \rho_2)^*$$

Since h is the height above the water table it can be seen that the initial values of capillary pressure vary vertically but not horizontally.

The relationship between capillary pressure and water saturation has been experimentally determined by

* Symbols are defined in the appendix

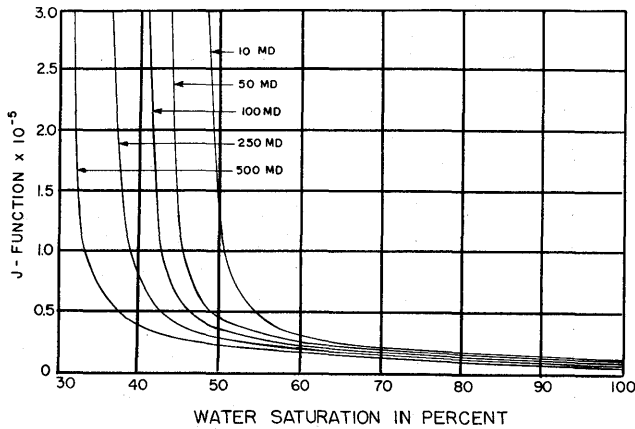


Figure 5—Effect of permeability on the J-function

Rose and Bruce.¹⁰ This relationship is given by the equation proposed by Leverett.¹¹

$$J(S_w) = (P_c/\sigma) (K/\phi)^{1/2}$$

Since K is permeability and ϕ is porosity, the quantity $(K/\phi)^{1/2}$ may be different for each computational module, but it will not vary with time. σ is interfacial tension between the liquids which does not vary.

Experimental values for the function $J(S_w)$ exist in the form:

$$A_{i,j} = (J(S_w) - B_{i,j}) (S_{w,i,j} - C_{i,j})$$

where i and j are indexes corresponding to a particular computational module. $A_{i,j}$, $B_{i,j}$, and $C_{i,j}$ are given for each computational module, since they depend on rock properties. To obtain a value of $J(S_w)$ it is only necessary to substitute a value of S_w into the relationship.

$$P_c = (J(S_w)\sigma) / (K/\phi)^{1/2}$$

Rose and Bruce¹⁰ illustrated the nature of the capillary retention curves, referred to as $J(S_w)$ functions. It is evident that there is no universal curve, but even though the curves vary with rock type, they each have the general shape of a hyperbola. The coefficients $A_{i,j}$, $B_{i,j}$, and $C_{i,j}$ control the shape of the hyperbola. These coefficients are determined from porosity, permeability and the type of rock formation. Since there are significant differences in the correlation of the J -function with water saturation from formation to formation no universal curve can be obtained. Correlation of the J -function with water saturation for a number of different materials is shown in Figures 5 and 6. The effects of permeability when all other parameters are held constant is shown in Figure 5. Figure 6 illustrates the effects of the other parameters when permeability

is held constant. These curves compare favorably with those of Rose and Bruce.¹⁰

RELATIVE PERMEABILITY

In the discussion of permeability the concept was restricted to rock property. This assumes that the pore space of the rock is completely saturated with a single fluid. This is called the absolute permeability of the rock. When more than one liquid is present in the pore space of a rock there is an effective permeability associated with each liquid present.

The relative permeability is calculated from the Corey equations.¹² For oil the equation is

$$K_{rn} = [1 - ((S_w - S_{LR}) / (S_m - S_{LR}))]^2 \times [1 - ((S_w - S_{LR}) / (1 - S_{LR}))]^2$$

The equation for water is

$$K_{rw} = [(S_w - S_{LR}) / (1 - S_{LR})]^4$$

where S_m is a constant which varies with formation type and is supplied as an input parameter, S_{LR} is the minimum water saturation and is a property of each computational module. Its value is the ordinate intercept of the 'asymptote' of the J -function.

The product of the absolute and relative permeabilities is called the effective permeability. This product divided by viscosity of the liquid is the transmissibility of the liquid.

Figure 7 shows a typical plot of oil and water relative permeability curves for a particular rock as a function of water saturation.

Starting at complete water saturation, the curves

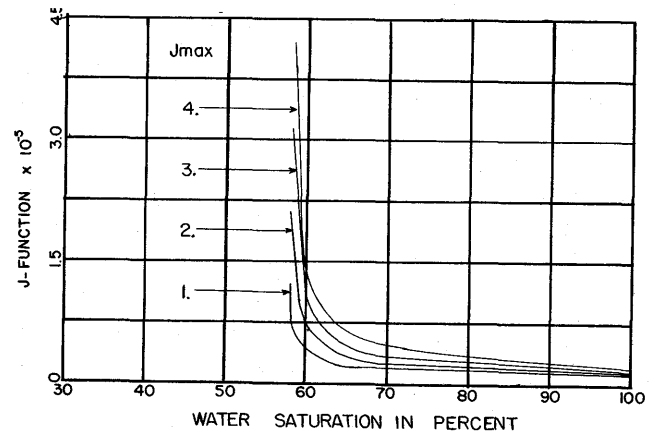


Figure 6—Effect of J-function asymptote on capillary pressure curves

show there is a decrease of eighty-five percent down to fifty-five percent.

At seventeen percent oil saturation, the relative permeability to oil is essentially zero. This value of oil saturation, seventeen percent in this case, is called the critical saturation. This is the saturation at which oil will first begin to flow as the oil saturation increases. It is also called the residual saturation, the value below which the saturation cannot be reduced in an oil-water system. This is why all of the oil in a formation cannot be recovered. When the water saturation in the production module is increased above its residual water saturation water begins to flow.

ANALYSIS OF RESERVOIRS

No matter how complete the coring and precise the data, one is still limited to an examination and study of rock samples which can constitute at the most an extremely small fraction of the total reservoir volume. This sample may be of the order of one ten-thousandth of one percent of the total reservoir.⁸ This small areal sampling of a reservoir could suffice for a description of the gross average properties of the producing formation. On the other hand, the ultimate limitation imposed thereby on the quantitative applicability of core-analysis data cannot be ignored. The basic fact is that all the features of the rock which are measured in core analyses are often so variable in passing from sample to sample along the well bore that the exact numerical data for a single sample are of little importance. What are significant are the average values for a set of neighboring samples or the large differences between

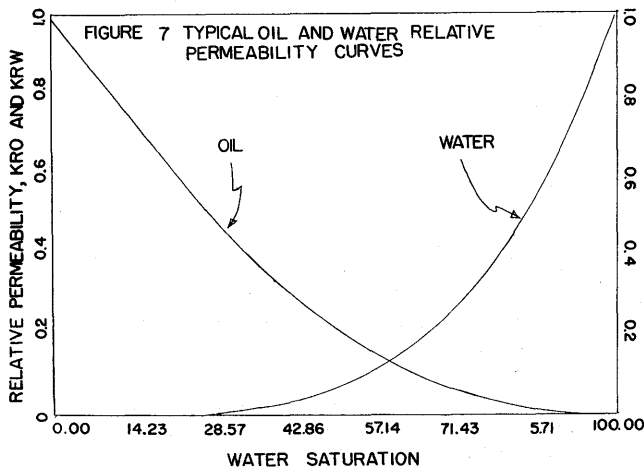


Figure 7—Typical oil and water relative permeability curves

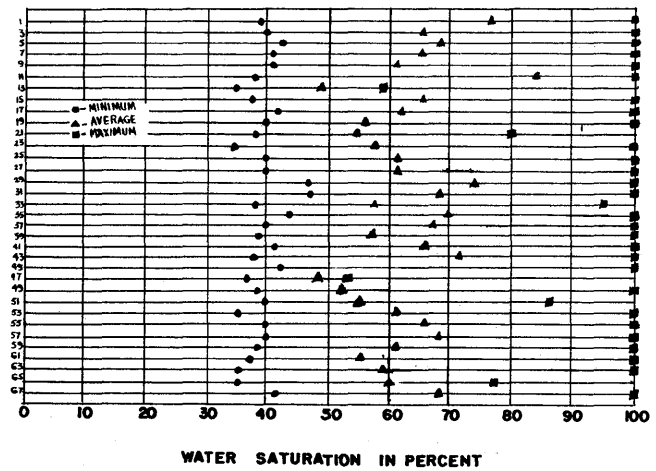


Figure 8—Minimum, average, and maximum water saturations from 70 wells—Ten feet of sand

adjacent groups of samples, which may indicate changes in type of strata or transition zones with respect to fluid content.

An appreciation of the concepts of porosity, permeability, and fluid saturations may be crystallized in terms of the numerical values associated with these terms. Unfortunately, however, no well-defined set of “typical” magnitudes can be given for these quantities. These quantities not only vary from formation to formation, but also from well to well in the same geologic stratum. Even in a single well, while penetrating a particular zone, the variations in the actual core-analysis data from sample to sample may be so large that simple averaging over the whole section may be unjustified and the supposedly single stratum must be considered as a composite of several distinct rock layers.⁸ Indeed, it is much easier to exhibit the variability in core analysis data than to provide average results of any significance.

This paper takes cognizance of the variability in rock properties from point to point. This is accomplished by combining rock simulation, capillary pressure, fluid properties, and physical laws to yield a representation of geologic strata which illustrates a typical reservoir.

Three of the properties which have been analyzed are water saturation, permeability and porosity. These properties were generated from samples on seventy consecutive wells using the procedures described previously. The samples were analyzed in one-foot increments for a sand thickness of ten feet located fifty feet above the water table.

Data pertaining to the water saturation is shown in Figure 8. This figure shows the minimum, average, and maximum water saturations for the odd numbered

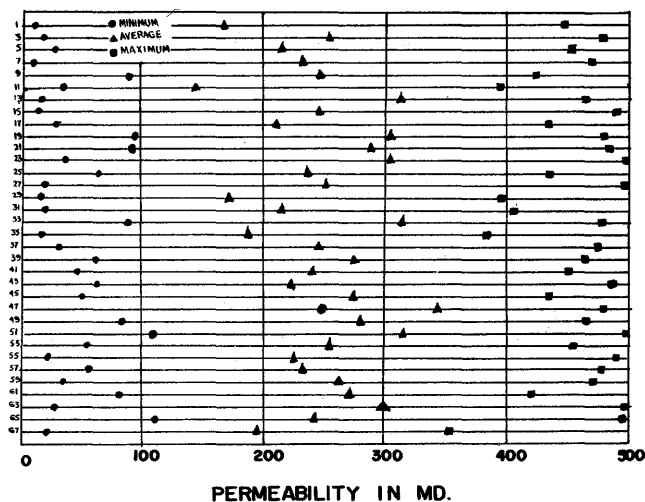


Figure 9—Minimum, average, and maximum permeabilities from 70 wells—Ten feet of sand

wells. The minimum water saturations are distributed between thirty-five and forty-five percent while the maximum water saturation observed in at least one sample from a single well was one hundred percent for more than eighty-five percent of the wells. It should be noted, however, that one well has a minimum value of approximately thirty-eight percent water saturation while its maximum value is only fifty-three percent. This fact, combined with the realization that the average water saturation is between forty-eight and eighty-two percent, reasserts the futility of utilizing an average core sample to represent an entire reservoir.

Figure 9 illustrates the variability of permeability in a sample of the previous seventy wells. As shown, the minimum permeability in a single well can range from a minimum of ten millidarcys to one hundred and ten millidarcys while the maximum values vary between three hundred and sixty and five hundred millidarcys. More interesting, perhaps, is the range of the average permeability. It varies from one hundred and fifty to three hundred and fifty millidarcys. Again, the fallacy of using average values from core analysis of one well is graphically illustrated.

Concluding this study of the core analysis of seventy wells is the display of fraction porosity values shown in Figure 10. This figure seems to display a tighter distribution of values than Figures 8 and 9, but it should be remembered that the range of the porosity is between 0.1 and 0.35. Since the range is smaller, minimum values between 0.14 and 0.202 contrasted with maximum values between 0.25 and 0.32 are not as clearly defined as one might expect. The average values seem to fill in

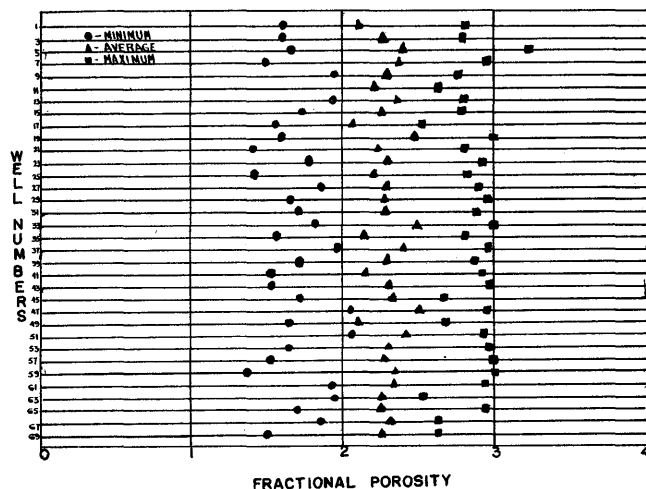


Figure 10—Minimum, average, and maximum porosities from 70 wells—Ten feet of sand

the gap as they range from 0.21 to 0.31. These values seem especially appropriate to illustrate that an average value in one well may be a maximum or a minimum for another well in the same reservoir.

CONCLUSIONS

This paper demonstrates the feasibility of simulating heterogeneous permeable strata for numerical study on high speed computers. The method uses the actual core data for permeability and porosity. The porosity is related to the permeability by a distribution curve utilizing a random number generator. The resulting fluid saturations for each foot of rock may then be computed by using a relation between capillary pressure, rock properties and fluid saturations when drilling and coring permeable strata.

REFERENCES

- 1 J E BRIGGS T N DIXON
Some practical considerations in the numerical solution of two-dimensional reservoir problems
Soc of Pet Engrs Jour June 1968
- 2 J DOUGLAS JR D W PEACEMAN
H H RACHFORD
A method for calculating multi-dimensional immiscible displacement
Trans AIME 1959 Vol 216 297
- 3 J BJORDAMMEN K H COATS
Comparison of alternating direction and successive overrelaxation techniques in simulation of reservoir fluid flow
Soc of Pet Engrs Jour March 1969

4 J BJORDAMMEN
Comparison of three methods for simulating two- and three-dimensional flow in reservoirs
 Master Thesis The University of Texas at Austin January 1968

5 IBM
System/360 scientific subroutine package
 Version III Programmers Manual No H 20-0205-3
 New York 1968

6 K D TOCHER
The art of simulation
 The English Universities Press LTD London 1963

7 B C CRAFT M F HAWKINS
Applied petroleum reservoir engineering
 Prentice Hall Inc New Jersey 1959

8 MORRIS MUSKAT
Physical principles of oil production
 McGraw Hill Book Co Inc New York 1948

9 J W AMYX D M BASS JR R L WHITING
Petroleum reservoir engineering
 McGraw-Hill Book Company New York New York 1960

10 W ROSE W A BRUCE
Evaluation of capillary character in petroleum reservoir rock
 Trans AIME 1949 Vol 186 127

11 M C LEVERETT
Capillary behavior in porous solids
 Trans AIME 1941 Vol 142 152-169

12 C E JOHNSON JR
Graphical determination of the constants in the Corey equation for gas-oil relative permeability ratio
 Journal of Petroleum Technology October 1968

APPENDIX

Nomenclature

<u>Symbol</u>	<u>Definition</u>
P_c	Capillary Pressure, psi
S	Saturation
h	Verticle Position Measured Positively Downward, feet
K	Absolute permeability, darcy
K_r	Relative permeability
Σ	Standard Deviation for Porosity
ρ	Density, psi/ft.
σ	Interfacial tension, dynes/cm.
ϕ	Porosity
<u>Subscripts</u>	
c	Capillary
i	Index for numbering blocks in the X-direction
j	Index for numbering blocks in the Z-direction
LR	Minimum water saturation
rw	Relative to wetting phase
rn	Relative to non-wetting phase
n	Non-wetting phase
w	Wetting phase

Digital simulation of the general atmospheric circulation using a very dense grid

by W. E. LANGLOIS*

Notre Dame University
Notre Dame, Indiana

INTRODUCTION

The dynamics of the weather represents, in its full generality, a computational problem which far exceeds the capability of any computer presently foreseeable. Fortunately, however, specific aspects of the weather problem can be profitably attacked with computers already in existence.

The large-scale motion of the atmosphere, usually termed the *general circulation*, is one such aspect. Computationally, it is a digital simulation problem based on a spatial finite-difference grid which, from an anthropocentric point of view, is rather coarse. A general circulation research model with a grid-spacing of 1° of longitude by 1° of latitude (110 kilometers by 110 kilometers at the equator) would be regarded as having extremely high resolution—unrealistically high for present-day computers.

One must bear in mind, however, that even the $2\frac{1}{2}^\circ$ by 2° coverage used in the present investigation distributes 12816 grid points over the surface of the earth. Thus the large-scale wind systems, which have horizontal length scales of 1000 kilometers or more, are not grossly underresolved. Sub-grid-scale effects are important, to be sure, but their *details* are only weakly coupled to the large-scale motion. For example, the grid is far too coarse to resolve the dynamics of a single cumulus cloud, but the net effect of cumulus activity in a grid cell can be reasonably well parameterized in terms of the grid-scale quantities.

Because of the complexity of general circulation calculations, $2\frac{1}{2}^\circ$ by 2° global coverage is feasible only with computers in the class of the IBM 360/91 or the CDC 7600. During the 1960s, general circulation research was carried out with much coarser resolution, $5^\circ \times 4^\circ$ being typically considered a "fine grid" (hence we have termed $2\frac{1}{2}^\circ$ by 2° resolution the "hyperfine

grid"). Nevertheless the 1960s produced significant advances in understanding the general circulation. As the decade progressed, the various models (summarily described by Kolsky¹) began to simulate the main features of the circulation rather well. Certain important details, to be discussed below, do require high resolution, and of course these are the focus of current interest, but the principal permanent and semi-permanent circulation systems can be realistically modeled with a $5^\circ \times 4^\circ$ grid.

The above discussion pertains only to *research* models of the general circulation, not to *forecast* models—for which the resolution problem is quite different. This may seem paradoxical since, after all, there is only one general circulation. Presumably it is governed by the same dynamical system, whether we are trying to understand its behavior or to forecast its evolution from an observed initial state. If an infinitely fast computer were available (actually a million MIPS might be enough) there would in fact be no distinction between a research model and a forecast model. Realistically, however, each type of model must leave off certain features of the other type in order to retain those features essential to its intended purpose. A research model requires global (or at least hemispherical) coverage, representation of the non-adiabatic atmospheric processes, and very-long-term simulation. A forecast model requires dense coverage, initialization from observed data, and near-real-time operation. The incompatibility of these requirements is illustrated by the fact that the $2\frac{1}{2}^\circ \times 2^\circ$ simulation reported here requires $2\frac{1}{2}$ hours of CPU time (about 6 hours of real time) per simulated day with the program running in a 1000 kilobyte partition of an IBM 360/91, even though the vertical resolution is quite coarse (2 vertical levels).

TWO SPECIAL ACKNOWLEDGMENTS

The general circulation model used in the present study is a version of that developed at UCLA by A.

* Visiting Professor of Mathematics

Arakawa and Y. Mintz, with the collaboration of A. Katayama. It was at Professor Arakawa's suggestion, and under his guidance, that the hyperfine grid simulation was undertaken. Working from the UCLA listings, H. C. W. Kwok and the author reprogrammed the model to run efficiently on a "pipeline" computer. A few minor thermodynamic modifications were incorporated but most of the model's physics remains as outlined by Arakawa, Mintz, and Katayama in their Tokyo paper.² A detailed description of the physics, and of our final code, is available in the series of reports by Langlois and Kwok.³

Since preparation of the paper in which we used the $5^\circ \times 4^\circ$ version of the model to study air contaminant transport,⁴ Kwok has transferred to projects not concerned with general circulation research. Fortunately for the present author, however, he had already solved most of the formidable data-management problems associated with hyperfine grid simulation.

DESCRIPTION OF THE MODEL

What follows is a reasonably complete, but entirely verbal, description of the general circulation model. The mathematical details are available in our reports,³ except for certain aspects of the radiation model and cumulus parameterization which are described in the appendices of the paper by Arakawa, Mintz and Katayama.²

The model troposphere is divided into two quasi-horizontal layers of equal mass. Specifically " σ coordinates" are used, i.e., the vertical coordinate is

$$\sigma = (p - p_t) / (p_s - p_t),$$

where p_s is the surface pressure and p_t is the tropopause pressure which is taken to be a constant 200 millibars. Thus the upper tropospheric layer corresponds to $0 \leq \sigma \leq \frac{1}{2}$ and the lower layer to $\frac{1}{2} \leq \sigma \leq 1$. The earth's surface, which follows the elevation of the large-scale mountain systems, corresponds to $\sigma = 1$. Vertical differencing is carried out in a way which conserves the first and second moments of potential temperature, as well as other physical quantities obeying integral conservation laws.

Horizontal differencing is carried out in the longitude-latitude plane. In this plane the image of the earth's surface is a rectangle of height π and width 2π . Except in the immediate vicinity of the poles, the finite-difference grid is constructed by subdividing this rectangle into a network of congruent rectangular cells measuring $2\frac{1}{2}^\circ$ of longitude by 2° of latitude. Near each pole, one row of grid points is skipped. Thus the grid cells along the 90° N or S latitude lines correspond, on the surface

of the earth, to $2\frac{1}{2}^\circ$ wedges extending from the pole to 87° latitude. The motivation for skipping points near the poles is linear stability: At 89° latitude the $2\frac{1}{2}^\circ$ longitudinal spacing corresponds to only 5 kilometers, which would require far too short a time step. The convergence of meridians near the poles is further mitigated by an averaging procedure which tends to damp out short waves moving in the longitudinal direction. Except for these details, the space-differencing is based on Arakawa's conservative differencing scheme, derived from the dynamical equations written in flux form.

A time step of $2\frac{1}{4}$ simulated minutes is used. The time differencing scheme is a variation of the Matsuno two-stage scheme, which approximates backward differencing. It differs from Matsuno's original scheme in two particulars:

- (1) The non-adiabatic processes are not calculated at every time step, since they are relatively slowly varying (time scale about one hour); to over-resolve them would greatly slow up the simulation.
- (2) The fluxes are not estimated the same way at all time steps, nor at both stages of the same time step. "Checkerboard instability" is avoided by alternating between centered and uncentered estimates.

The surface underlying each grid cell is specified as being ice-free ocean, sea ice, ice-free land, glacier, or snow-covered land. Grid cells corresponding to ice-free ocean are assigned surface temperatures appropriate for the season; since the present paper describes a simulation of northern hemisphere winter, the observed January values are used. Land surface is regarded as a thermal insulator with no capacity to store heat; its temperature is determined by balancing incoming and outgoing thermal fluxes. If the land is ice or snow covered, this temperature is constrained not to exceed the melting point of ice. Sea ice is treated like ice-covered land, except that there is some heat conduction thru the ice. The ice distribution is specified for northern hemisphere winter; the snow distribution is estimated as a function of calendar date.

The dependent variables are the surface pressure, the temperatures and horizontal wind velocities of the two layers (the vertical differencing scheme represents these as carried at $\sigma = \frac{1}{4}$ and $\sigma = \frac{3}{4}$), and the mixing ratio of the lower layer. Since the moisture-carrying capacity of the air diminishes rapidly with decreasing temperature, and hence with increasing altitude, the mixing ratio is carried rather low, viz, at $\sigma = \frac{1}{8}$. The moisture content of the relatively cold upper layer is neglected. However, in the final section we present some

evidence that this approximation should really be modified at hyperfine grid resolution.

The model accounts for four contributions to the non-adiabatic heating and cooling: incoming solar radiation, mostly visible and near infra-red, which depends on latitude, season, and local time of day; long-wave infra-red radiation (usually a heat sink); sensible heat transfer between the lower layer and the underlying surface; release of latent heat during precipitation.

Radiative heating and cooling of the air in a grid cell depend on the mixing ratio and on the nature and extent of the cloud cover. Three types of clouds are distinguished: stratus deck, which results from large-scale condensation occurring when the mixing ratio exceeds its saturation value; cumulus towers, which result either from convection in the middle portion of the troposphere or from penetrating convection originating in the planetary boundary layer; low-level cumulus clouds, which result from boundary layer convection that is too weak to penetrate into the middle troposphere, and which produce no rain.

The moisture source in the model is evaporation from the open sea, from ice or snow covered surfaces, and from ice-free land that has previously been moistened by rain. The evaporation rate depends on the surface wind speed and on the vapor pressure difference between the air and the surface. For ocean, ice, and snow, the surface vapor pressure is the saturation value for the surface temperature; for ice-free land it depends on the history of precipitation, evaporation, and runoff.

MOTIVATION FOR HYPERFINE GRID SIMULATION

As indicated in the introduction, the main features of the general circulation can be simulated without resort to hyperfine grid resolution. This can be seen, for example, from the pressure maps in the GARP study,⁵ which employed a $5^\circ \times 4^\circ$ version of the UCLA model (the "fine grid" version described in our reports.³ However, certain aspects of weather *development* are not well simulated with a $5^\circ \times 4^\circ$ grid. For example:

1. The west to east progression of cyclonic storms is too slow—about 5° per day, whereas 10° , or slightly more, is typical of the real atmosphere.
2. The development of storms is likewise too slow.
3. The subtropical highs are 5 to 10 millibars too weak. This is related to the previous two items: cyclonic eddies form the principal mechanism for removing heat from the subtropics and, with this process slowed down, thermal lows tend to weaken the highs.

Manabe, Smagorinsky, Holloway, and Stone⁶ described a high-resolution simulation using the hemispherical general circulation model developed at the Geophysical Fluid Dynamics Laboratory of the Environmental Sciences Services Administration. Their horizontal differencing uses a uniformly spaced grid on a stereographic projection centered at the pole, with 40 grid points between pole and equator. On the average, this yields horizontal resolution roughly comparable with that of the present study.

The GFDL group compared the results of their high-resolution simulation with those of a previous study in which they used 20 grid points between pole and equator (roughly comparable to our $5^\circ \times 4^\circ$ grid). They reported that the system of fronts and the associated cyclone families in the high resolution model are much more realistic than those of the low-resolution model. Moreover, they analyzed the energetics of the simulation in some detail, finding that the general magnitude and the spectral distribution of kinetic energy are in better agreement with the actual atmosphere as a result of the improvement in resolution. These findings offer further incentives for hyperfine grid resolution with the UCLA model. It must be borne in mind that the two models accept computer limitations in entirely different ways. For example, the GFDL model assumes a featureless earth, but uses nine levels of vertical resolution. However, the improvements reported by the GFDL group appear to result from better horizontal resolution of the major transport mechanisms essentially common to both models.

THE SIMULATION

As implied in the introduction, research models of the general circulation are not usually initialized from real data. Rather, the model generates its own data which, in present day models, is statistically reasonable but which does not correspond in detail to the weather on any actual day.

When a model is run for the first time, its "cold start" is taken from an artificial initial state. For example, we use a neutrally-stable and completely dry atmosphere at rest with a uniform sea-level air temperature of 0° centigrade. This choice is easily programmed even for a model with mountains, and it offers the additional advantage that realistic patterns evolve after only two simulated weeks (cold start from an isothermal atmosphere requires about three times as long to achieve realism because of the extreme stability). Subsequent runs are initialized from a history tape whose records are generated at specified update intervals (usually 6 simulated hours). There is

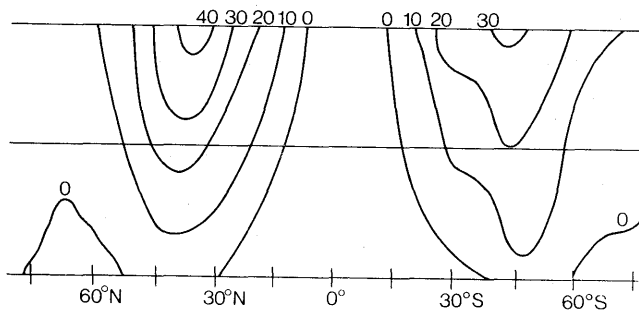


Figure 1A—Zonal mean westerlies after 30 days of fine-grid simulation

also provision for generating a history tape record at the end of a run which is interrupted between regular updates.

To save computer time, we did not run the hyperfine grid simulation from cold start. Instead, we generated a history tape by interpolating the final state of a 20 day simulation with the $5^\circ \times 4^\circ$ grid. This preparatory simulation began from cold start with climatic data appropriate to Nov. 1.

The first 10 days of hyperfine grid simulation were carried out with the non-adiabatic processes calculated every twenty time steps (45 simulated minutes). At this point we compared the static features of the simulation with those of a 30 day run with the $5^\circ \times 4^\circ$ grid.

The distributions of the mean zonal temperatures for the two cases were substantially the same. The permanent and semi-permanent features of the sea-level pressure maps differed primarily in that the Azores and Siberian highs were about 10 millibars more intense with the hyperfine grid. Changes in the wind pattern are more pronounced; Figure 1 compares the mean zonal westerlies for the two cases. The vertical structure was obtained by linear interpolation in σ from the computed values at $\sigma = 1/4$ and $\sigma = 3/4$. The most notice-

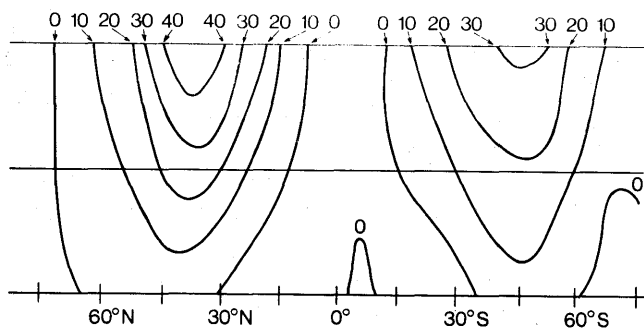


Figure 1B—Zonal mean westerlies after 20 days of fine-grid simulation followed by 10 days of hyperfine grid simulation

able features of the hyperfine grid result are the intensification of the northern hemisphere jet stream and the appearance of a core of weak westerlies in the inter-tropical convergence zone.

These first ten days of hyperfine grid simulation revealed that carrying all the moisture in the lower layer is not fully acceptable at $2\frac{1}{2}^\circ \times 2^\circ$ resolution. The problem arises when a grid cell undergoes low-level convergence and high-level divergence in a region of high relative humidity. In nature this produces dynamically forced convective rain over the grid cell, with the released latent heat being divided between the σ -layers, and with some outward water vapor transport in the upper layer. In the model, however, the upper layer is incapable of carrying moisture. Hence the model sees large amounts of water vapor advected into the lower layer of the grid cell, but none carried upward in spite of the pronounced lifting. This leaves the lower layer grossly supersaturated. Intense large-scale condensation then takes place and the concomitant release of latent heat—all in the lower layer—unrealistically destabilizes the atmosphere.

This effect was discovered because of an oversight in setting up the hyperfine-grid run. Passing to higher resolution involves changing the continental outlines. As illustrated in Figure 2, certain areas which repre-

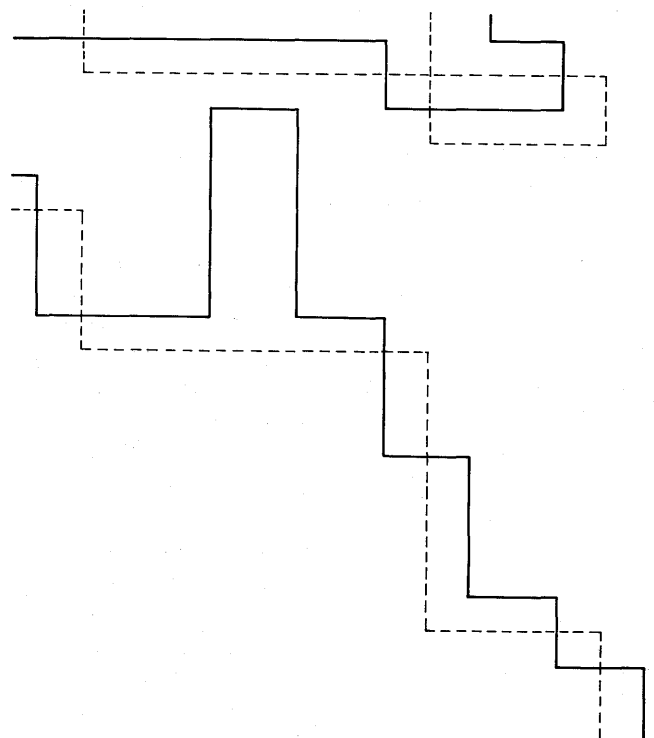


Figure 2—Resolution of the Coral Sea shorelines: dotted line, fine grid; solid line, hyperfine grid

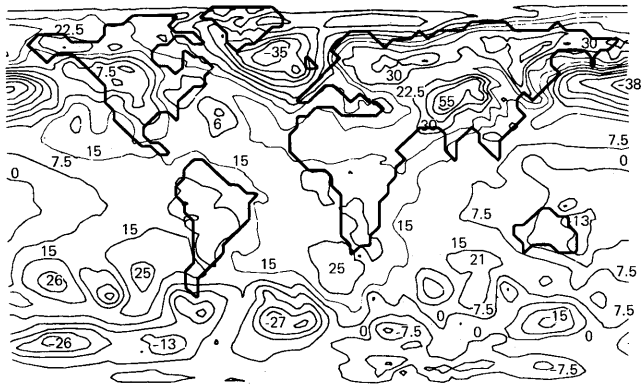
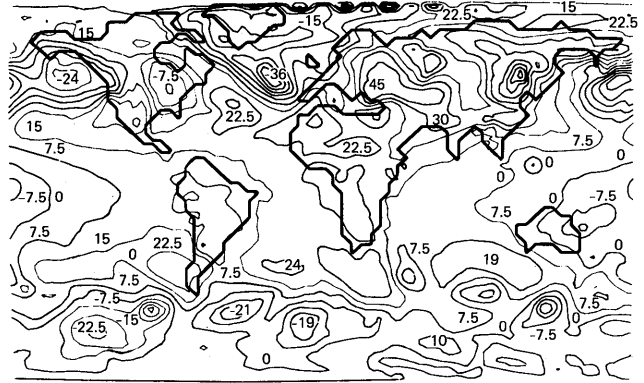


Figure 3—The simulated sea-level pressure
A. Day 29



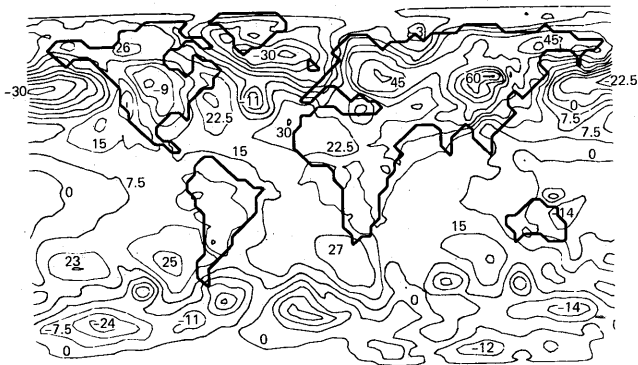
C. Day 31

sented coastal regions of Australia or New Guinea with $5^\circ \times 4^\circ$ resolution represent parts of the Coral Sea when the grid is refined to $2\frac{1}{2}^\circ \times 2^\circ$. The hyperfine-grid history tape was generated by interpolating data from the fine-grid tape at midnite GMT of "November 20", i.e., at 10 A.M. Brisbane time during southern hemisphere summer. Consequently the surface temperatures of New Guinea and Queensland were rather high (35 to 45 degrees centigrade). Thus some of the Coral Sea hyperfine-grid cells were assigned surface temperatures far too high for ocean. Prodigious evaporation immediately took place. A center of convergence just off the Queensland coast completed the requirements for runaway precipitation, causing a local hot spot. The simulation was continued with the ocean temperatures cooled off to a maximum of 304° Kelvin, which relieved the Coral Sea problem. However, unrealistically high supersaturation at a few grid points continued to result from the model's inability to simulate dynamically forced convection. Since the algorithm for computing the large-scale condensation is a second-order Newton-

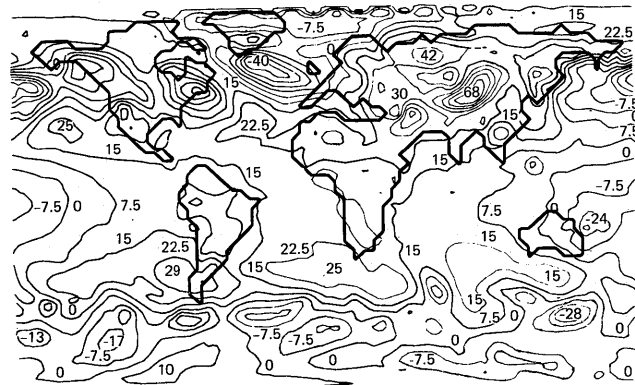
Raphson procedure it appeared advisable to avoid trouble by artificially removing moisture in excess of 140 percent relative humidity. Since this grossly unrealistic supersaturation typically occurs at only 1 to 7 grid points out of nearly 13 thousand, the artificial disruption of the model's moisture balance is insignificant.

During the final 10 days of the simulation, the interval for calculating the non-adiabatic processes was reduced to ten time steps ($22\frac{1}{2}$ simulated minutes). The objective of this was to eliminate more smoothly the unrealistic results of the sea-temperature error. In retrospect, however, this was probably unnecessary: short comparison runs reveal that changing this interval from 20 to 10 time steps produces hardly any discernible effect.

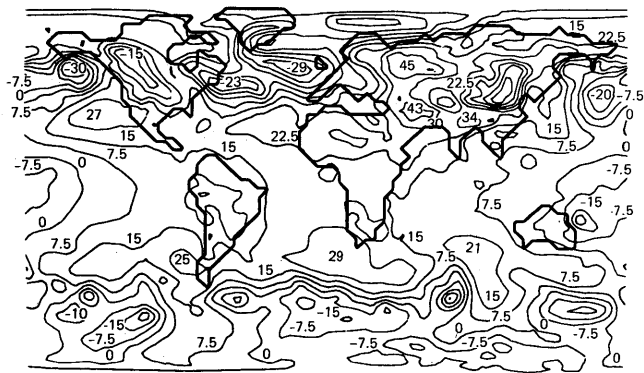
The sequence of simulated events during the last twelve days of the simulation is depicted in Figures 3A thru 3L. These are maps of the surface pressure reduced to sea level. The contour interval is 7.5 millibars and



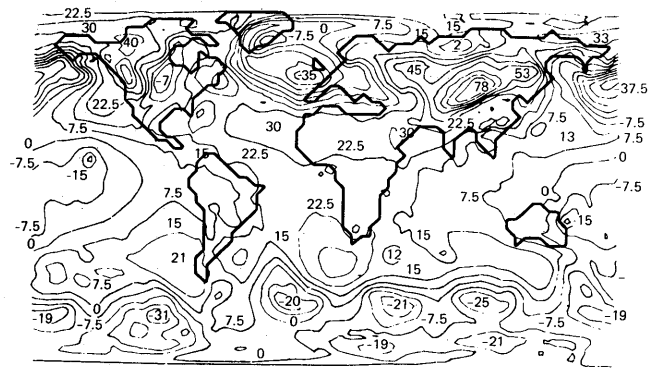
B. Day 30



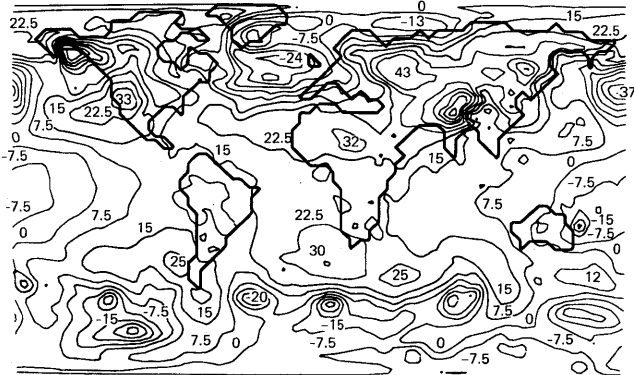
D. Day 32



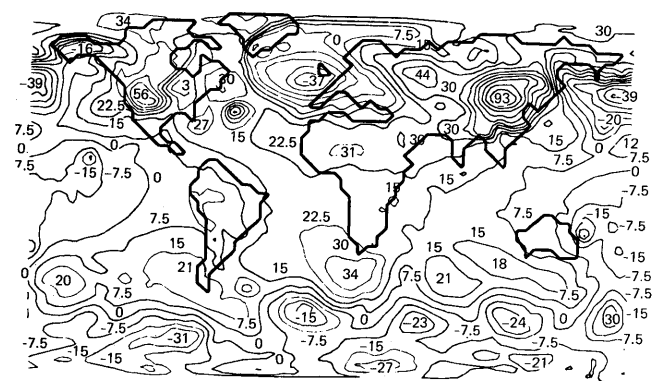
E. Day 33



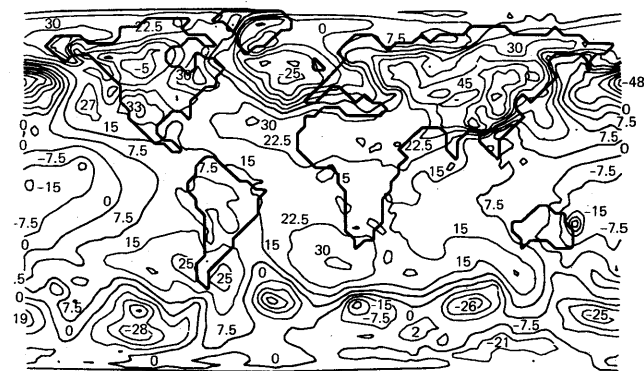
H. Day 36



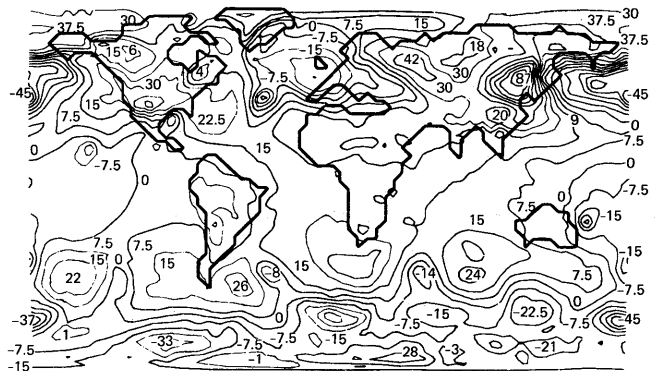
F. Day 34



I. Day 37



G. Day 35



J. Day 38

the numerical data indicate pressure excess over 1000 millibars. In each case the simulated time is 00:00 GMT of the indicated day. The thermal low over the Coral Sea should be ignored because it resulted from the runaway precipitation described above.

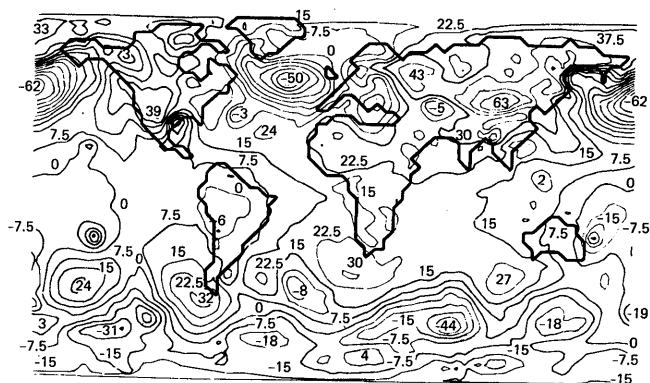
The success of hyperfine grid resolution in simulating the motion and development of cyclonic storms is best seen from this sequence by examining the activity over North America and the eastern North Atlantic. The history of four separate storms can easily be followed from the twelve maps in Figure 3.

On Day 29 a broad 1006 mb low was situated off the New England coast. A day later it had intensified to 989 mb and moved 15° east. By Day 31 it had moved 10° more and merged with the Icelandic low.

Also on Day 31, a 992 mb low appeared over Hudson Bay. It rapidly intensified to 884 and one day later it was 16° to the east over northern Quebec. Its rate of progression then slowed to 9° per day and on Day 34 its merger with the Icelandic low was in progress.

On Day 35 a 995 mb low appeared over the Mackenzie district. This one intensified only slightly (to 993 mb), then weakened as it passed across Hudson Bay and over the Hudson Strait. During its three-day life span it moved eastward 29°.

The last three maps depict rather complicated cyclonic activity over the United States. The movement and development seem to be heavily influenced by the

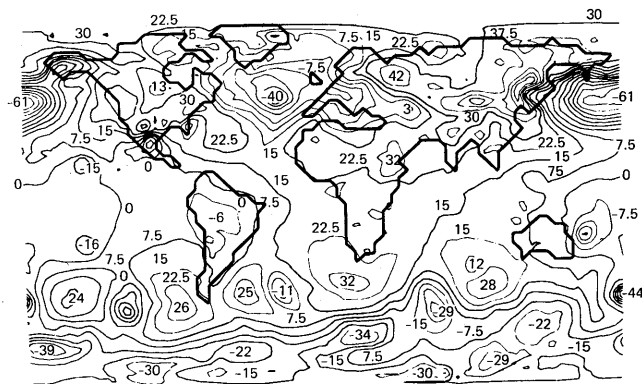


L. Day 40

blocking effect of a pair of high pressure regions over southern Canada.

REFERENCES

- 1 H G KOLSKY
Some computer aspects of meteorology
IBM Journal of Research and Development 11 584 1967
- 2 A ARAKAWA Y MINTZ A KATAYAMA
Numerical simulation of the general circulation of the atmosphere
Proc of the WMO/IUGG Symposium on Numerical Weather Prediction Tokyo Sect IV p 7 1968
- 3 W E LANGLOIS H C W KWOK
Numerical simulation of weather and climate
A series of reports of the Large-Scale Scientific Computations Department IBM Research Laboratory San Jose California
I. *Physical description of the model 1969*
II. *Computational aspects 1969*
III. *Hyperfine grid with improved hydrological cycle 1970*
- 4 H C W KWOK W E LANGLOIS R A ELLEFSEN
Digital simulation of the global transport of carbon monoxide
IBM Journal of Research and Development 15 p 2 1971
- 5 R JASTROW M HALEM
Simulation studies related to GARP
Bulletin American Meteorological Society 51 p 490 1970
- 6 S MANABE J SMAGORINSKY
J L HOLLOWAY JR H M STONE
Simulated climatology of a general circulation model with a hydrologic cycle III. Effects of increased horizontal computational resolution
Monthly Weather Review 98 p 175 1970



K. Day 39

Simulation of the dynamics of air and water pollution

by LAURENCE W. ROSS

University of Denver
Denver, Colorado

INTRODUCTION

Simulation of the dynamics of air and water pollution rests firmly on the *diffusion equation*, which in simplest form is known as Fick's second law. The problem of dispersion of solutes and suspensoids is much older than the pollution crisis, and in fact the development of the first useful solutions of the diffusion equation came in response to a need for predicting the spread of poison gas.

Between the World Wars, a small group of English investigators pressed forward steadily with this development. The first really useful result came in 1923, and in 1932 Sutton developed the three-parameter formula that is still the most widely used in the regulations of several states, and in the gas dispersion correlations of the U.S. Army Chemical Corps. The state of development up till about 1950 is admirably summarized in Sutton's text.¹

The decade of the 1960s witnessed a very strong upsurge in the development of dynamic models for dispersion of air and water pollution. The opening of the 1970s finds the federal government committed to the rational management of all our natural resources, and thus we see a strong upsurge in development of air pollution models for urban environments, especially. In the realm of water pollution, *thermal pollution* poses an immediate threat that has defied realistic simulation, so far, and here we also observe a great upsurge in activity. The problem of solid waste pollution is a problem in systems engineering, not simulation, and we must neglect it here even though it has strong potential for environmental crisis in the 1970s.

Despite the massive efforts, we observe very few fundamental advances. There are good reasons for the paucity of really useful, successful simulation models of air and water transport, and that is the subject of this paper.

AIR POLLUTION

General theory

The transport of particulates and gases in the lower atmosphere is influenced by all the natural mechanisms that give rise to motion. In general, therefore, the correct mathematical description of the atmospheric environment must include the rate expressions that govern the three conserved quantities of the physical world: momentum, energy, and mass. These general rate equations are shown in Table I.

In air pollution, the equations of energy do not enter the mathematical description except at definite points (e.g., stacks) where energy is added to the system. Therefore, the energy equation of Table I may reasonably be neglected on the large scale; it will reappear when we consider behavior of smoke plumes.

Therefore, the usual set of equations for description of air pollution dynamics is the following:

Equation of continuity
Equations of motion
Equation of diffusion.

The diffusion equation is obviously coupled to the equations of motion by the velocity terms. The set is usually further simplified by assigning

$$u = u(z) \\ v = 0 \quad \text{or} \quad w = 0.$$

It will be observed that if $v = w = 0$, then $u = \{\text{constant}\}$. Therefore, one other velocity besides the horizontal must be retained if the altitude dependence of u is to be retained.

On a large scale, it is often convenient to retain v and permit the wind to possess two components parallel to

TABLE I—The Equations of Transport

Momentum	
x-direction:	$\rho \left(\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} + v \frac{\partial u}{\partial y} + w \frac{\partial u}{\partial z} \right) = -\frac{\partial P}{\partial x} - \left(\frac{\partial \tau_{xx}}{\partial x} + \frac{\partial \tau_{yx}}{\partial y} + \frac{\partial \tau_{zx}}{\partial z} \right)$
y-direction:	$\rho \left(\frac{\partial v}{\partial t} + u \frac{\partial v}{\partial x} + v \frac{\partial v}{\partial y} + w \frac{\partial v}{\partial z} \right) = -\frac{\partial P}{\partial y} - \left(\frac{\partial \tau_{xy}}{\partial x} + \frac{\partial \tau_{yy}}{\partial y} + \frac{\partial \tau_{zy}}{\partial z} \right)$
z-direction:	$\rho \left(\frac{\partial w}{\partial t} + u \frac{\partial w}{\partial x} + v \frac{\partial w}{\partial y} + w \frac{\partial w}{\partial z} \right) = -\frac{\partial P}{\partial z} - \left(\frac{\partial \tau_{xz}}{\partial x} + \frac{\partial \tau_{yz}}{\partial y} + \frac{\partial \tau_{zz}}{\partial z} \right) + \rho g_z$
Energy:	$\rho C_p \left(\frac{\partial T}{\partial t} + u \frac{\partial T}{\partial x} + v \frac{\partial T}{\partial y} + w \frac{\partial T}{\partial z} \right) = k \left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} + \frac{\partial^2 T}{\partial z^2} \right)$
Mass:	$\frac{\partial C}{\partial t} + u \frac{\partial C}{\partial x} + v \frac{\partial C}{\partial y} + w \frac{\partial C}{\partial z} = \left(D_x \frac{\partial^2 C}{\partial x^2} + D_y \frac{\partial^2 C}{\partial y^2} + D_z \frac{\partial^2 C}{\partial z^2} \right)$
Continuity:	$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = 0$
Assumptions:	Constant density (ρ), thermal diffusivity ($k/\rho C_p$), and mass diffusivity (D_i), and absence of viscous dissipation contributions to energy.

the earth. This is the procedure adopted by Hino.² On an intermediate scale (\sim miles), the coordinate system is often defined such that $w=v=0$, and a mean horizontal wind velocity \bar{u} is used. On the very smallest scale (\sim 1000 yards), a velocity profile is assumed *a priori*; the theory of turbulent flow usually leads to adoption of the Prandtl mixing-length model (see for example Randerson³):

$$\kappa u/u_* = \ln(z/z_0) + \text{const.}$$

Alternatively, a power-law approximation is sometimes used for convenience in computation, or in similarity solutions.

The diffusion coefficients present a different sort of theoretical problem, because they must *always* be modeled. Quite often D_x is neglected in view of the assumption

$$u(\partial C/\partial x) \gg D_x(\partial^2 C/\partial x^2).$$

The lateral and vertical diffusion coefficients, D_y and D_z , have therefore received principal attention, especially D_z . However, the best results are still quite empirical (see Pasquill).⁴

At this point, it is convenient to mention the *Richardson number*, which is defined as

$$Ri = \frac{(g/T_0)[(dT/dz) + \Gamma]}{(du/dz)^2}.$$

The Richardson number may be regarded as the ratio of the thermal driving force for vertical air motion to the vertical force arising from turbulent shear. It is the basic parameter for micrometeorological motion, in the

same sense that the Reynolds' number is basic to confined fluid motion. For example, the vertical diffusion coefficient D_z is often expressed as a function of Ri as follows:

$$D_z = \kappa^2 z^2 (du/dz) (1 - \sigma Ri)^\beta$$

where β is usually taken as $1/2$. On a small or intermediate scale, where the diffusion coefficient cannot be averaged, the Richardson number may have to be known.*

At this point, we have reduced the diffusion equation to the form:

$$\frac{\partial C}{\partial t} = \frac{\partial}{\partial y} \left(D_y \frac{\partial C}{\partial y} \right) + \frac{\partial}{\partial z} \left(D_z \frac{\partial C}{\partial z} \right) - \bar{u} \frac{\partial C}{\partial x} - R. \quad (1)$$

The transient equation is practically never of interest, and R is rarely considered (of this, more below), and the remaining equation has the following general solution (for a point source of emission):

$$C = \frac{Q}{\pi \bar{u} \sigma_y \sigma_z} \exp \left[-\frac{1}{2} \left(\frac{y^2}{\sigma_y^2} + \frac{z^2}{\sigma_z^2} \right) \right]. \quad (2)$$

Particular solutions, based upon particular choices of the form of σ_y and σ_z , are shown in Table II. The Bosanquet-Pearson solution, for example, carries the implicit assumption that⁵

$$D_y \propto \bar{u} x$$

$$D_z \propto \bar{u} z.$$

* Note that, in general, $Ri = Ri(z)$.

Modeling urban air pollution

Equation (2) is the basis of most current air pollution models for urban areas. There have been numerous applications (see Tikvart⁶ and Stern⁷), differing mainly in the treatment of the source field, assignment of σ_y and σ_z , and whether or not short-range wind structure is analyzed. It should be noted that the scale of air pollution dispersion suggests the use of the asymptotic form of equation (2), viz.,

$$C = Q / \pi \bar{u} \sigma_y \sigma_z. \tag{3}$$

Miller and Holzworth⁸ are the principal exponents of this approach. Bowne⁹ has reported a digital simulation of air pollution patterns over the State of Connecticut, based on Equation (2). Hino² solved a coupled system involving the two-dimensional diffusion equation and the equations of motion (Navier-Stokes) in two dimensions to handle the problem of topographical variations. Interestingly, the grid square dimensions are usually either 1 mile or 1 km in all investigations.

The basis of the urban modeling method, when using Equations (2) or (3), is to employ experimental data as a

TABLE II—Practical Solutions of the Diffusion Equation for Air Pollution (Case of Continuous Point Source in a Wind)

1. Sutton equation

$$\begin{aligned} \sigma_y/x &= (1/\sqrt{2}) C_y x^{-n/2}, \quad \sigma_z/x = (1/\sqrt{2}) C_z x^{-n/2} \\ C &= (2Q/\pi C_y C_z u_x x^{2-n}) \\ &\times \exp \{ -x^{n-2} [(y^2/C_y^2) + (z^2/C_z^2)] \} \end{aligned} \tag{II.1}$$

Note: $n = 1/4$ for neutral atmosphere, $1/5$ for strong lapse, $1/2$ for strong inversion

2. Calder equation

$$\begin{aligned} \sigma_y/x &= \sqrt{2} (aku_*)/u_x, \quad u_z = \sqrt{2} (ku_*)/u_x \\ C &= (Qu_x/2k^2 au_*^2 x^2) \exp \{ -(\mu_z/ku_* x) [(y/a) - z] \} \end{aligned} \tag{II.2}$$

Note: Not Gaussian!

3. Bosanquet-Pearson equation

$$\begin{aligned} \sigma_y/x &= q, \quad \sigma_z/x = \sqrt{2} p \\ C &= (Q/2^{3/2} \pi u_x p q x^2) \exp [-(y^2/2q^2 x^2) - (h/p x)] \end{aligned} \tag{II.3}$$

4. Modified Sutton equation

$$\begin{aligned} & n_y - 1 \qquad \qquad \qquad n_z - 1 \\ \sigma_y/x &= \sqrt{2} C_y x \qquad , \quad \sigma_z/x = \sqrt{2} C_z x \\ C &= (Q/2\pi C_y C_z u_x x^{n_y + n_z}) \exp [-1/2 (y^2/C_y^2 x^{2n_y})] \end{aligned} \tag{II.4}$$

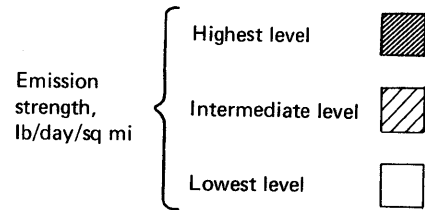
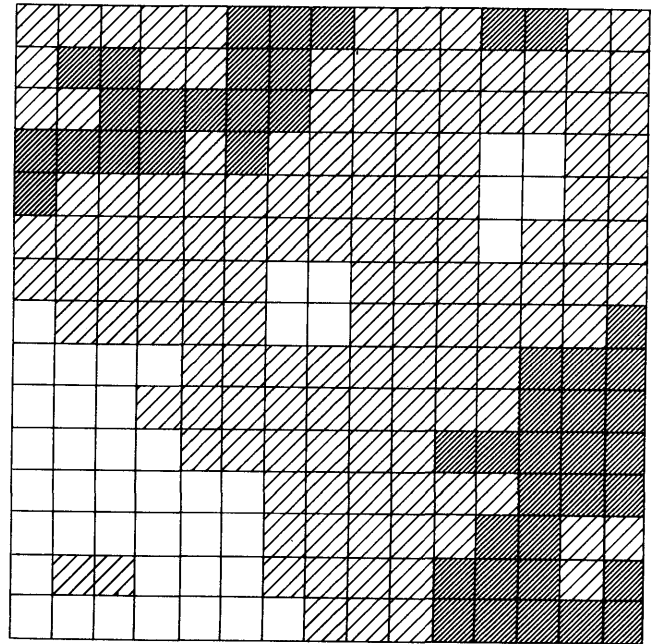


Figure 1—Typical pattern of pollution emission strengths. Typical grid dimension is 1 mile square

means of assigning each grid square as a point source of given emission strength (Figure 1). It is wasteful of computer storage to maintain more than a few levels of emission strength in the model (e.g., three levels in Figure 1). Then the mean wind speed at a given direction is applied, together with assignments of σ_y and σ_z based upon the model chosen, and concentration in a given cell is computed as the resultant of the contributions from other cells, above some predetermined lower limit of concentration. Figure 2 illustrates the principle.

Day-to-day pollution control requires a somewhat different approach. Here the problem is to produce the pollution pattern (as in Figure 1) from moment-to-moment measured data, then to apply (2) or other predictor relation to obtain estimates of pollution levels. Figure 3 shows the situation with respect to data monitoring stations. With distance and position of emission sources established with respect to wind direction, it is feasible to estimate the concentration of

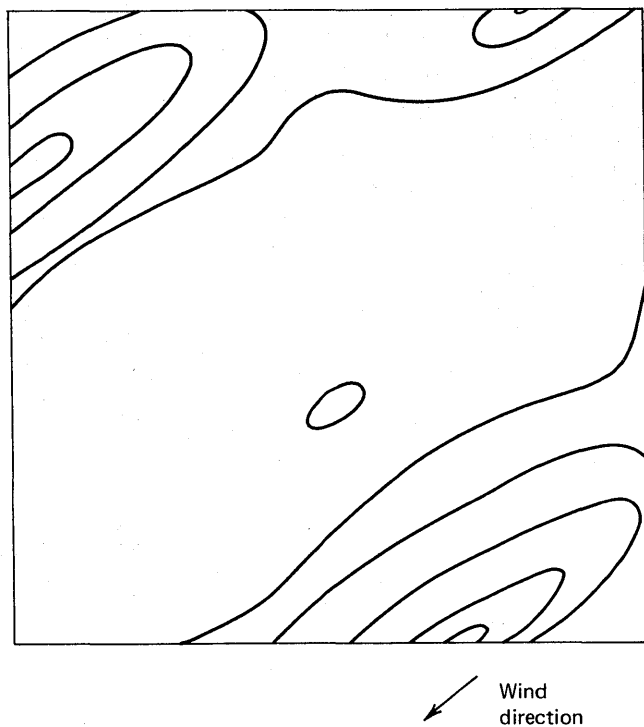


Figure 2—Pollution isopleths resulting from applying dispersion model to measured emission patterns

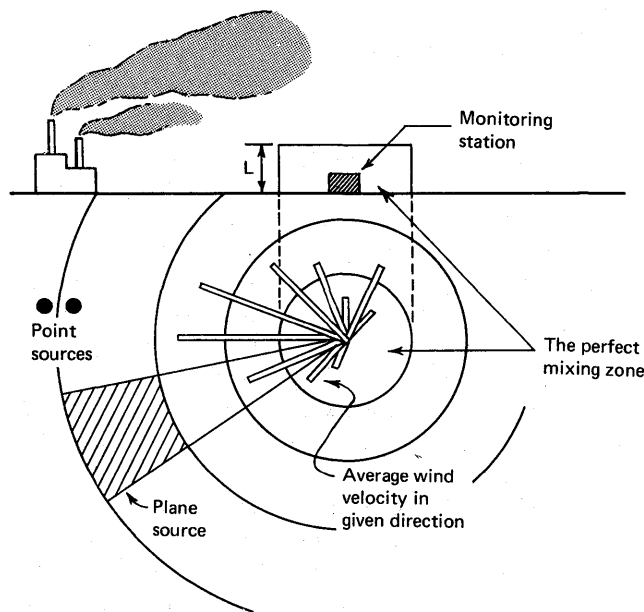


Figure 3—Features of an emission monitoring system for air pollution control. It should be noted that individual sources are identified

a given pollutant at any position, if its stoichiometry is known. For example, the use of fuel of known sulfur content will produce a given amount of SO₂. This is the model described by Takamatsu *et al.*,¹⁰ for air pollution control in Osaka. Furthermore, Takamatsu's model considers a smaller scale than that of Bowne⁹ or Randerson,³ and recognizes that the region near ground level is subject to different meteorological patterns than higher regions. This leads to consideration of a separate layer, the "complete mixing zone," where the diffusion equation applies, the zone above being assumed a perfect sink for pollutants. This also resembles the "box model" of Lettau.¹¹

The so-called complete mixing zone has not been identified formally with the famous "inversion layer," nor is the APCO definition of "mixing depth" the same as the depth of this zone, necessarily. Indeed, the failure to simulate inversion effects is one of the principal embarrassments of simulation efforts to date. By whatever definition, it is clear that a criterion for a layer of finite depth is required. The only theoretical basis for such a finite depth is the *stability height* of Monin and Obukhov,¹² formalized in 1954 as

$$L = - \frac{u_*^3}{\kappa(g/T_0)(q/c_p\rho)} \quad (4)$$

This is essentially equivalent to a normalization of height against a Richardson-number criterion. It requires a knowledge of friction velocity u_* and heat flux g , but this is not excessively demanding and the present author believes that the stability height deserves more use.

Special cases

Although "special," these may be the cases of more intense public interest. For example, the simulation of *smog dynamics* is obviously unsatisfactory, for otherwise the means of smog control would have found their way into legislation instantly. The reaction kinetics of smog processes has been deciphered with reasonable confidence, but the physical influences—diurnal temperatures and winds, air-water interactions, etc.—are still mysteries. For example: What is the ultimate fate of smog? No one knows.

The dynamics of *smoke plumes* has received intensive study, and seems to be well understood (see for example References 13, 14). This case is especially interesting because it requires simultaneous consideration of the equations of momentum, energy, and mass (Table I). Practically no simplifications are available in the general case, and this becomes a demanding exercise in

computer simulation. Another feature of the smoke plume problem is that it is a *natural convection* problem, which fact invites the application of two-dimensional analysis by combination of variables and computation of stream functions; this approach has not been applied, to date.

Future directions for air pollution simulation

The basic missing ingredient in air pollution simulation is *meteorological measurement*. Emission measurements, on the other hand, are fairly well advanced, except that we still have not had the political courage to pinpoint individual sources of strong emissions.* This lack of measurement is surprising, because it would be simple, and a series of brilliant experiments by English scientists have provided ample verification of the theory. The quantities that require measurement are not in doubt.

The current programs in various cities provide masses of data for regression analysis. However, these obviously have application only locally, and only then for limited periods of time. Thus, they cannot be used for prediction except in a statistical sense, and they are useless for control. To rectify this situation, it is merely necessary to obtain data on the same basis that the theoretical models are constructed. This calls for wind speed and direction (at other points than just the local airport!), and the wind and temperature profiles in vertical direction. The "stability depth" must also be established, but the other measurements will probably make this automatic.

Chemical change in the atmosphere has received very little consideration up to the present. Most simulation experiments have been based on sulfur dioxide, overlooking the fact that about three-fourths of the SO_2 emitted to the atmosphere is converted to H_2SO_4 which is rapidly removed by condensation. Thus (for example), a downwind variation will probably yield inaccurate conclusions about the diffusion coefficients, because reaction is also a significant mechanism of pollutant elimination. Smog pollution has prompted some important studies of reaction kinetics, but the complexity of smog reactions and the physical influences on smog (moisture, sunlight, mountain barriers, etc.) make this a very difficult subject. Thus, we are in the unfortunate position of lumping chemical changes into diffusion coefficients, which leads us to conclude that we must obtain diffusion coefficients for each separate polluting species. Most important of all, mechanistic models of diffusion coefficients become meaningless.

* The Japanese, to their credit, have done this.

Reaction in the atmosphere should logically be simulated by supplying a functional form for R , the reaction rate. In the case of SO_2 and CO , this form may be satisfied by a first-order assumption, i.e.,

$$R = -k_R \cdot C.$$

The literature contains a few values for k_{SO_2} , but they vary over two orders of magnitude, which probably points to the influence of moisture, associated fly ash, sunlight, and possibly ozone. There is practically nothing available for other gases. Alternatively, the disappearance of gaseous species may be simulated by a *sink* term (constant) in the diffusion equation, but no investigators have reported this method.

In the case of particulates, there is loss by deposition. This generally calls for inclusion of $v(\partial C/\partial z)$ in the diffusion equation, and a suitable boundary condition at the earth's surface, e.g.,

$$\lim C(x, z) = (Q/u)\delta(z)$$

as suggested by Calder.¹⁵

Smoke plume simulation is seldom the subject of computer simulation. Nevertheless, plumes have been studied extensively, because of their importance in prediction of pollution from stacks. The situation is very complex, combining the influences of fluid motion, energy, and diffusion, so that analytical solutions are not reasonable to seek. Csanady^{13,14} is the outstanding investigator in the field; there is a useful review of the subject by Brummage.¹⁶ A typical mathematical formulation of this situation for the case of vertical plumes is given by

$$\begin{aligned} \frac{\partial}{\partial z}(\rho r w) + \frac{\partial}{\partial r}(\rho r u) &= 0 \\ \frac{\partial}{\partial r}(\rho r w^2) + \frac{\partial}{\partial r}(\rho r u w) &= r \left(\frac{\theta'}{\theta_e} \right) \rho g + \frac{\partial}{\partial r}(r \tau) \\ \frac{\partial}{\partial z}(\rho r w \theta) + \frac{\partial}{\partial r}(\rho r u \theta) &= -\frac{1}{C_p} \frac{\partial}{\partial r}(r F) \end{aligned} \quad (5)$$

When written in cylindrical coordinates, which is natural for vertical plumes, the use of analog computer techniques is possible (see for example Reference 17).

SIMULATION OF WATER POLLUTION DYNAMICS

General theory

The dispersion of pollutants in water is identical to dispersion in air, at least in principle. The general equations of transport (Table I) still apply.

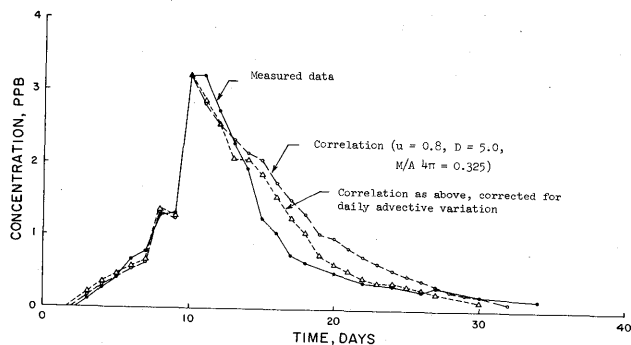


Figure 4—Dispersion of pollutant in the Potomac River as function of time, 2.3 miles downstream of injection point¹⁷

However, measurements in waterways are somewhat more difficult to obtain than measurements in air, and we find that the diffusion equation is universally written as

$$\frac{\partial C}{\partial t} = D \frac{\partial^2 C}{\partial x^2} - \bar{u} \frac{\partial C}{\partial x} \quad (6)$$

O'Connor¹⁸ seems to be the only investigator who has applied two-dimensional modeling, although numerous authors have recognized that the general model must be multi-dimensional. The usual one-dimensional expression obviously lumps lateral and vertical dispersion effects into the longitudinal parameters D and \bar{u} . Furthermore, the velocity \bar{u} is usually assigned as the overall average,

$$\bar{u} = Q/A$$

so that the burden on D is all the greater.

Despite the theoretical reservations that must surround such a simplified model, it has been remarkably successful. The advantages of the model—two parameters plus Gaussian form—are considerable, because waterways do exhibit this form of behavior (Figure 4).¹⁹

The model according to Equation (6) may describe either pollutant, expressed as BOD or COD, or dissolved oxygen (DO). In the case of DO, a suitable boundary condition is required, usually

$$\left\{ \frac{dC}{dz} = k_L(C^* - C) \right\}_{z=0}$$

Sometimes, the right side of this expression is added to Equation (6), implying that aeration of the waterway is a homogeneous process; this is incorrect.

Equation (6) often requires addition of a reaction term R . This may describe either reactive decay of the dissolved pollutant or "dead zones" describing imperfect mixing, as defined by Krenkel.²⁰

Lumped-parameter simulation

The simulation of real streams presents several problems, especially those of tributary influx and the mainstream velocity variation (meander). This has led several investigators to consider that simulation by lumped-parameter formulations is required, in order to absorb all the distorting influences. The method has been used for many years by individual industries to describe dispersion of their pollutant discharges, but the definitive formulation is given by Thomann²¹ in the following form:

$$\begin{aligned} V_i(\partial C_i / \partial t) = & Q_i[\xi_i C_{i-1} + (1 - \xi_i) C_i] - Q_{i+1}[\xi_{i+1} C_i \\ & + (1 - \xi_{i+1}) C_{i+1}] + E_i(C_{i-1} - C_i) \\ & + E_{i+1}(C_i - C_{i+1}) + k V_i C_i + P_i \end{aligned} \quad (7)$$

A good example of the application of this method is reported by Hetling.²² The method is obviously suitable for analog simulation if the parameters are available, or if the data are sufficient to permit extraction of the parameters by potentiometer twiddling or formal methods.²³

The lumped-parameter method of simulation can always be made to succeed if (and only if) the data supply is sufficient to permit evaluation of the parameters. This is the great virtue of the lumped-parameter method. On the other hand, the method contributes nothing to theory, and the parameters cannot be extended to other waterways or even to different situations on the same waterway.

Simulation of thermal pollution

Thermal pollution has emerged as a major problem because our waterways will soon be saturated with heat, if the present rate of growth is maintained. Nuclear power plants are especially serious offenders in terms of waste heat.

The dispersion of waste heat in waterways requires consideration of the energy equation (Table I). In lakes or ponds, or in well-behaved waterways, the simulation may be based on straightforward application of the energy equation, viz.,

$$\partial T / \partial t = \nabla^2 \cdot DT - u(\partial T / \partial x) \quad (8)$$

However, the boundary conditions of thermal pollution are difficult, because they must include the effects of radiation, conduction, and evaporation at the waterway surface:

$$\begin{aligned} \left\{ -D(\partial T / \partial z) = h_r(T_0^4 - T^4) \right. \\ \left. + h_c(T_0 - T) + k_e(p^* - p) \right\}_{z=0} \end{aligned}$$

The difficulty of defining waterway velocity, in all but the simplest situations, has discouraged the use of this formulation. Edinger²⁴ has used a linearized version, lumping all boundary effects into a single coefficient of exchange, for the case of a cooling pond. The nonlinear radiation term may be avoided by simply specifying the radiative flux, which may be taken as constant over a given period.

Simulation is usually achieved by lumped parameters. Jaske²⁵ has the principal body of work here, but Yearsley's work²⁶ best represents the current thinking of the federal water establishment. The federal government has published a manual that suggests modeling of thermal pollution by a simple first-order decay law.

Modeling of thermal pollution by two-layer representations had some early success, but has been neglected in recent years. The principle, obviously, is based on considering a hot layer atop a cold layer; this is a very reasonable model, as we have observed in the laboratory. The only basic difficulty with the concept is the necessity to describe the shear stress at the two-layer interface, but this should be capable of extraction by well-known methods if sufficient data are available.

Estuarine salinity models

The dynamics of salinity exchange in estuaries is not exactly water pollution dynamics, but is interesting because (1) salinity exchange is a problem of differential densities, similar to thermal pollution, and (2) some very fine work has been performed in this field probably foreshadowing future developments in water pollution.

For example, the development that Ratray²⁷ uses in describing steady-state circulation in fjords is as follows:

$$u \frac{\partial u}{\partial x} + w \frac{\partial u}{\partial z} = - \frac{1}{\rho} \frac{\partial}{\partial x} (p + \frac{1}{2} \rho u^2) + \frac{\partial}{\partial z} \left(A \frac{\partial u}{\partial z} \right) - \frac{\partial p}{\partial z} + \rho g = 0$$

$$\frac{\partial}{\partial x} (ub) + \frac{\partial}{\partial z} (wb) = 0$$

$$u \frac{\partial S}{\partial x} + w \frac{\partial S}{\partial z} = \frac{\partial}{\partial z} \left(D_z \frac{\partial S}{\partial z} \right) \quad (9)$$

Still required is a relation between density and salinity, $\rho(S)$ usually taken as a linear function. Then introduction of stream functions and combined variables yields (as usual) a nonlinear set that can readily be

resolved by computer methods. The number of parameters is formidable, but the results are promising.

Future directions for water pollution simulation

Techniques of water pollution simulation are essentially at a standstill. This fact reflects the rapidly improving water pollution situation, and the difficulty of improving upon existing two- and three-parameter models.

Thermal pollution is the one major area where the problems are growing rapidly, and this is where simulation will probably be needed soonest. The available correlations are not adequate, and recourse to methods based upon the energy equation seems certain, sooner or later.

We should probably expect no improvement in the theory, in the foreseeable future. In contrast to the atmosphere, which is vast enough to be described by more or less general theory, waterways are highly individualistic. The problem of meander is fundamental, and it will resist simulation for some time to come, until the need for understanding of our crowded waterways on a short-range scale provides the stimulus.

CONCLUSIONS

Simulation of air and water pollution dynamics has developed rapidly, but will probably experience no outstanding developments in the 1970s. The theoretical basis is satisfactory, and the current generation of computers is entirely adequate for the task of simulation.

The chief restrictions on the development of this field are those of *measurement*. Neither air pollution nor water pollution measurement programs, as presently implemented in the U.S., provide sufficient data for control or for generalizations that may be applied to the nation's urban areas.

In air pollution, the simulation of urban pollution patterns (especially smog patterns) is moving steadily forward. However, the present author is convinced that several basic considerations are being neglected, which could easily be repaired by attention to the theory so painstakingly developed by foreign investigators. It is difficult to resist the conclusion that political considerations outweigh the scientific considerations.

In water pollution, the theory ends with two-parameter Gaussian dispersion models. All efforts past this point have resort to linear, lumped-parameter models. There is room for breakthroughs here, but there is no particular impetus for them, so we should not expect them in the 1970s. The one possible exception is

thermal pollution, which may become a crisis item in the 1970s.

Since Japanese cities have long since resorted to fuel consumption regulation based upon modeling of air pollution dynamics, it seems very logical to expect similar considerations to be applied in the U.S. in the foreseeable future, in both air and water pollution.

REFERENCES

- 1 O G SUTTON
Micrometeorology
McGraw-Hill New York 1953
- 2 M HINO
Atmos Environment
2 541 1968
- 3 D RANDERSON
Atmos Environment
4 615 1970
- 4 F PASQUILL
Atmospheric diffusion
Van Nostrand New York 1962
- 5 A I DENISOV
Izvest Akad Nauk SSSR Ser Geofiz
6 834 1957
- 6 J A TIKVART
Computer simulation and air quality control
Paper Published by NAPCA 1970
- 7 A C STERN Ed.
Proceedings of Symposium on Multiple-Source Urban
Diffusion Models U S Environmental Protection
Agency APCO Publ No AP-86 1970
- 8 M E MILLER G C HOLZWORTH
Journal Air Pollution Control Assoc
17 46 1967 *ibid* 232
- 9 N E BOWNE
Journal Air Pollution Control Assoc
19 570 1969
- 10 T TAKAMATSU et al
Computer control system for air pollution
Published by Kyoto University and the Osaka
Prefectural Govt 1967
- 11 H H LETTAU
*Physical and meteorological basis for mathematical models
of urban diffusion processes*
In Stern, A C ed. Proceedings on Symposium on
Multiple-Source Urban Diffusion Models US EPA APCO
Publ No AP-86 pp 2-1 through 2-26 1970
- 12 A S MONIN A M OBUKHOV
Trudy Geofiz In-ta AN SSSR No 24 p 151 1954
- 13 G T CSANADY
Journal Applied Meteorology 10 36 1971
- 14 P R SLAWSON G T CSANADY
Journal Fluid Mech 47 33 1971
- 15 K L CALDER
Journal Meteorology 18 413 1961
- 16 K G BRUMMAGE
Atmos Environment 2 197 1968
- 17 M P MURGAI H W EMMONS
Journal Fluid Mech 8 611 1960
- 18 D J O'CONNOR
Journal San Eng Div ASCE 91 23 1965
- 19 L W ROSS
Simulation 14 95 1970
T SAVILLE
Bulletin No 125 Florida Eng and Ind Exp Sta
August 1966
- 20 J R HAYS P A KRENKEL
Advances in water quality improvement
Vol 1 p 111 Univ of Texas Press Austin 1968
- 21 R V THOMANN
Journal San Eng Div ASCE 89 No SA5 1 1963
- 22 L J HETLING R L O'CONNELL
Water Resources Research 2 825 1966
- 23 E S LEE I WANG
Journal Water Pollution Control Fed 43 306 1971
- 24 J E EDINGER J C GEYER
Journal San Eng Div ASCE 94 611 1968
- 25 R T JASKE J L SPURGEON
Water Research 2 777 1968
- 26 J YEARSLEY
*A mathematical model for predicting temperatures in rivers
and river-run reservoirs*
Working Paper No 65 US Dept of Interior FWPCA
March 1969

APPENDIX

SYMBOLS

a	Parameter of Calder's equation (Table II)
A	Area of waterway cross section; vertical coefficient of turbulent velocity
b	Width of waterway
C	Concentration of pollutant species
C_i	Concentration of pollutant species in segment i (Equation (7))
C^*	Oxygen saturation concentration in water
C_p	Heat capacity
C_y, C_z	Diffusion-related coefficients of Sutton's equation (Table II)
D	Diffusion coefficient of energy or mass in water (x -direction)
D_x, D_y, D_z	Diffusion coefficients in respective directions
E_i	Turbulent exchange coefficient (Equation (7))
F	Heat flux function
g	Gravitational acceleration
h_c	Film coefficient of conduction
h_r	Film coefficient of radiation
k	Rate constant (Equation (7) only)
k_e	Evaporation film coefficient
k_L	Oxygenation film coefficient
k_R	Reaction rate constant
L	Monin-Obukhov stability height

n	Parameter (Table II)	\bar{u}	Mean velocity in horizontal direction
n_y, n_z	Parameters (Table II)	v	Velocity in lateral (y) direction
p	Partial pressure of water vapor; static hydraulic pressure; parameter of Bosanquet-Pearson (Table II)	w	Velocity in vertical (z) direction
p^*	Vapor pressure of water	v_i	Volume of segment i (Equation (7))
P_i	Source term, segment i (Equation (7))	x, y, z	Cartesian coordinates
q	Heat flux (Equation (4)); parameter of Bosanquet-Pearson equation (Table II)	z_0	Roughness height
Q	Rate of pollutant emission	β	Parameter
Q_i	Pollutant flux in segment i (Equation (7))	Γ	Adiabatic lapse rate
r	Radial dimension	δ	Dirac delta function
R	Reaction rate	κ	Karman constant
Ri	Richardson number	θ	Potential temperature
S	Salinity of waterway	θ'	Reference potential temperature
t	Time	θ_e	Equilibrium potential temperature
T	Temperature	ξ_i	Correction for flow in segment i (Equation (7))
T_0	Surface temperature	ρ	Density
u	Velocity in horizontal (x) direction	σ	Parameter
u_0	Mean velocity	σ_y, σ_z	Standard deviations in the respective directions
u_*	Friction velocity, $\sqrt{\tau_0/\rho}$	τ	Shear stress
		τ_0	Shear stress at ground level

Programming the war against water pollution

by DEXTER J. OLSEN

International Business Machines Corporation
Kingston, New York

INTRODUCTION

In every communications medium, today, including newspapers and magazines, television and radio, and even in personal conversations, we are constantly alerted to the problems of pollution. More often than not, we are asked to *do* something about these problems. Most of us are learning of little things we can do individually, in our personal lives, to help reduce pollution.

As for the bigger things, we tend to sit back and say "they" should do something about them," and in saying "they," we are attempting to shift the burden to either our legislators or the proverbial "George." It is the intent of this paper to explore what we in the computer programming profession can do to assist in the alleviation of our water pollution problems.

Many of those who have entered the programming profession during the last few years, have frequently come directly from the college environment and increasingly are the products of specialized computer-science curricula. Our older colleagues, on the other hand, usually entered into programming as an outgrowth of, or even as an alternative to, other vocational backgrounds. They represent various degrees of experience in mathematics, chemistry, finance, business, physics, biology, engineering—the list is almost endless. It is this group of people that I especially want to address, because many of these disciplines that are represented among us can be applied in some manner to programming the war on water pollution.

It is not my intent to go very deeply into particular phases of the problem at hand, but rather, to cover the topic in a general way, talking about some of the things that have been done to date, and pointing out some of the things that have yet to be done, both in the way of technical items, and in bringing the computer closer to the engineer. It is hoped that in pointing out some of the problems involved, that a few sparks may ignite in some of your minds as to how your particular expertise, even

though seemingly not associated with water pollution, might be put to use, or at least combined with the expertise of others, to assist in this battle.

BACKGROUND

For years a few dedicated souls and a very few professions have been waging an uphill battle to try to interest the legislators and the general public in the need for action on this front. One of these professions has been civil engineering and, more particularly, the little known field of sanitary engineering. It is these engineers who are responsible, through their training, for the research and design of the facilities and structures which make the water we use fit for human needs, and to cleanse our waste waters to an acceptable tolerance before discharging them into a nearby river, stream or lake.

With all the importance being attached to this subject today and, after taking a look at the immense job that lies ahead, it is discouraging to find how little the electronic computer is being used in this battle.

Civil engineers were among the early users of computers for the more mathematized disciplines, such as: surveying, and the design of highways, bridges, and buildings. Only recently, however, have they begun to use computers in the areas requiring engineering judgment and decision-making. Even at that, many civil (and, especially, sanitary) engineers have been slow in utilizing computers. To a large extent this may be attributed to the fact that, unlike all too many federal projects, where large amounts of taxpayers' money always seem to be available, the sanitary engineer, until recently, has had to work with the rather limited funds from the local municipality, and, therefore, has felt that he must stay with the tried and proven methods of construction, treatment techniques and methods of design, rather than take chances with unproven innovations.

TABLE I—Volume of Construction Contracts (in millions of dollars)*

ITEM	ACTUAL				ESTIMATED			
	1967	1968	1969	1970	1971	1972	1975	1980
Sewerage	1,179	1,386	1,415	2,050	2,500	3,250	3,950	6,125
Waterworks	970	887	980	1,090	1,200	1,350	2,125	2,750
TOTALS	2,149	2,273	2,395	3,140	3,700	4,600	6,075	8,875

Another feeling still prevalent today among many of these engineers is that by turning their design problems over to a computer they will lose control of their designs and they will no longer bear their individualistic style. In spite of our daily exposure to it the computer is something that many others still cannot understand and many are afraid of. Fortunately these fears are being overcome slowly as the more adventurous engineering firms begin to use computers.

In this country there are approximately 70,000 civil engineers and only about 12,000 sanitary engineers. Many of these people work within very small consulting engineering firms that cannot afford the cost of any but the smallest of computers, or must resort to some form of time-shared computer use. Thus, the economic circumstances have also contributed to this slow acceptance of the computer. The advent of time-sharing systems is helping to alleviate this problem.

SIZING THE JOB AHEAD

We are all aware by now that one of the principal enemies of our country is pollution, and, according to some, we have barely begun to scratch the surface as to cleaning it up. But just how big is the job? The ultimate costs are difficult to determine and at best are educated guesses. It is a fact, however, that the volume of construction for public water and sewerage facilities, combined, in 1967 amounted to approximately \$2 billion and by 1970 had reached about \$3 billion. By 1980, this figure is expected to triple. At the beginning of 1971, the backlog of new construction planning for water and sewerage facilities was over \$16 billion (see Tables I and II). This latter figure represents in the neighborhood of \$400 million worth of actual design work, of which 50-60 percent could be done on computers if the proper programs were available.

This year the federal government, as well as many of the states, has taken action to make even more money available to local municipalities and sanitary districts. The construction trade journal *Engineering News-*

*Record** states that "for the next two years, sewerage construction will set the pace for gains in water use and control. The projected volume for 1972 is more than double 1969's dollar volume, and three and one-half times 1966's volume, yet it won't be nearly enough to bring river and stream pollution under control. Annual volume will have to redouble later in this decade to accomplish that goal."

TABLE II—Backlog of New Construction Planning (in millions of dollars)* as of December 31

ITEM	1967	1968	1969	1970
Sewage	7,635	8,029	9,487	12,399
Waterworks	4,152	3,983	4,077	4,019
TOTALS	11,787	12,012	13,564	16,418

* As compiled by *Engineering News-Record*, see bibliography.

One segment of the economy that has not been affected adversely during the recent recession has been the design and planning of anti-pollution facilities. All indications are that the volume of construction and the necessary design and planning in this area will be maintained at even higher levels over the foreseeable future.

DISSECTING THE JOB

Before we look at what the unprogrammed needs of the sanitary engineer are, we might do well to look over his shoulder to see what he is doing now. The range of problems he encounters might be categorized as follows:

1. Collection of data
2. Analyzation of the collected data
3. Population studies
4. Hydraulic studies

* "EN-R's 96th Annual Survey," *Engineering News-Record*, vol. 186, no. 3, page 23, January 21 1971.

5. Analyzation of existing facilities
6. Biological and chemical studies
7. Modeling of existing and proposed facilities
8. Hydrologic and water basin monitoring
9. Modeling of the body of water that will receive the treatment plant effluent
10. Implementing of design concepts
11. Structural design implementation
12. Selection of equipment to meet the various design criteria
13. Determination of operating costs
14. Process control
15. Planning and monitoring construction progress
16. Development of specifications
17. Estimating of construction costs
18. Development of detailed construction plans
19. Researching new treatment techniques.

Collecting and analyzing data

The data available in many instances today consists primarily of operating records of existing treatment plants. This data is adequate to meet the requirements of the state health departments, but is not suitable for providing meaningful information for use in designing new facilities or adding to existing plants. More frequent sampling and metering of incoming sewage and of effluent discharges, as well as more adequate sampling and metering of the receiving waters, are only some of the additional data required. Of course, with more data to collect and more data to analyze, adequate metering and telemetering systems must be made available, as well as sample-analysis systems. Means for adequately analyzing this increased volume of data must also be developed.

Population studies

Before analytical or design studies for any particular community can be started, population studies must be made. With all the census information available today, the problem of population studies is still quite a headache. In any given community, population figures must be broken down into quite a variety of districts for numerous functions, such as: school districts, voting districts, electric power districts, water districts, sewer districts, park districts, tax districts, and land zoning districts. In many instances, the boundaries of each of these types of districts do not coincide with any of the others. A population model or analyzation system would be much appreciated by the planners concerned with each of these endeavors.

Armed with the proper tools and information to establish the existing population within a given type of district, the analyst may then merge information pertaining to projected land use and other demographic data to determine the population totals for which he must design.

Modeling and simulation

A bare beginning has been made in the way of modeling existing and proposed treatment facilities. A look at some of the modeling attempts may suggest extensions that need to be done. Perhaps some of the modeling and simulation work that has been done in some other fields, such as chemical processes, might be adapted for use with water and waste treatment facilities.

Modeling of waterways

Models of flow in open channels, such as rivers, irrigation waterways, and flood control channels have principally concentrated on the flow, or even the dispersion of a pollutant, being distributed evenly over the cross section of the channel. Two-dimensional or even three-dimensional models need to be developed to help understand the mixing action of pollutants. What happens when these same pollutants are discharged into a river emptying into a tidal water or directly into salt water? There is known to be a "salinity wedge" that moves in and out of the mouth of a river with the tides and with the density of the constituents of the river water. What effects will this salinity wedge have on the dispersion of river water and its pollutants?

Little, if any, modeling or analysis has been done on the so-called "conservative" pollutants (chlorides, non-biodegradables, etc.) and their reactions, both biologically and chemically, and their mixing characteristics in a receiving body of water. Models of the biological and chemical reactions coupled in some manner with hydraulic models need to be developed. In addition, models may tell us what effect thermal pollution has on the biological, chemical and mixing characteristics of the river, lake or ocean. To make these models work, more and better data must be collected, which emphasizes once more the need for adequate data acquisition systems.

Another area in which models can be useful is as a guide in determining what data to collect, as well as where and how much to collect. In this way the value of the data may be determined prior to the collection effort and detailed modeling studies.

Modeling of treatment facilities

A start has been made in the modeling of waste treatment facilities which will help a consultant in determining the size and cost of certain treatment units. Much more needs to be done to cover the spectrum of possible treatment methods. Models to allow a consultant to determine the best and least costly type of treatment facilities for a particular municipality need to be developed.

A comprehensive computer model can assist the engineer in determining not only the best type of treatment facilities for a given situation, but can also help determine the environmental costs and benefits to the affected geographical region. For instance, many rivers serve as both a water supply source and a dumping ground for waste disposal for a series of communities. A model can assist engineers in studying the effects the water usage has on the various communities the river services, e.g., the costs, effects, and recreational benefits on farmland, lakes, streams, air, parks and general land use.

Models of the operating characteristics of a particular plant would be helpful not only during the design stages, but in helping to keep the plant operating efficiently during later years. Such model studies could be tied in to process control systems to run the plants. One of the first computer-controlled municipal sewage treatment plants is scheduled to go into operation in Nassau County, New York, during the fourth quarter of this year. Another is scheduled later for San Jose, California.

However, as with most "firsts," these probably will be rudimentary compared with those that will be constructed in the future. But two out of the thousands of possibilities across the country make it apparent that many more, with many more variations, must be devised.

Many communities have either no treatment facilities or have what is called "primary treatment" (gravitational the removal of only the readily settleable solids). As the various states establish more stringent anti-pollution requirements, more and more communities will be forced to go to more complex secondary or even tertiary treatment (processes providing different degrees of oxidation of the effluent resulting from the prior treatment), which becomes much more critical in their operating procedures. Computer control of these plants will soon become a must, and some say it is already a must.

Modeling of pipelines

With the ever increasing demands for more water, many of the larger cities are having to go greater dis-

tances than ever before to obtain adequate supplies. This means that long water transmission lines must be built to transport the water. Such a pipeline may require a number of pumping stations placed at intervals along its length. The designer must determine the number, frequency, and location of these pumping stations along with determining the size (diameter) of the pipeline. A computer model could combine these variables with the possible variation in the number of pumps on the line, the variable hourly demand for water, and the storage facilities at the city to determine the best arrangement of pumping stations and sizes of pipeline. Optimizing the design could save the taxpayers a lot of money. After completion of construction, computer control of the pumping stations could be employed to maintain operating efficiencies. I have been informed that very little programming has been done in this area.

The problem of "water hammer", or hydraulic transients, is of immense concern on large pipelines. A sudden surge of pressure is created by the sudden closing of a valve in the pipeline. In large pipes this surge of pressure against the valve and the pipe walls can reach amazing proportions. Being of a fluid nature, this pressure wave bounces from end to end of the pipe until it dampens out. The analysis of the water hammer problem is a highly complex and specialized field of study. As a result, "rule of thumb" designs often prevail and the pipeline is overdesigned. A computerized simulation of this phenomenon that could be used by the design engineer without a high degree of specialized training would be of great benefit. This could lead to better designs of pipelines, and less cost to the taxpayer.

Hydrologic monitoring

The monitoring of hydrologic and stream data for entire water basins is still in its infancy. The Tennessee Valley Authority and the Chicago Sanitary District have both established programs of telemetering this information in their respective watersheds. These data, when combined with meteorological data will enable us to design better and more adequate flood control facilities and may provide the basis for automatic control of the facilities themselves.

Problem-oriented languages—Structural and hydraulic design

Many programs have been written for structural analysis and design of bridges and buildings using structural steel shapes, but only a limited amount of programming has been done for underground structures

built of reinforced concrete, which is necessary for water and sewage treatment facilities. The designers need a problem-oriented language suitable for the non-computer-oriented engineer to design beams, columns, walls, cantilevered walls, troughs and a myriad of other shapes that are possible with poured concrete. All these must be coupled with application of various types of loadings, such as: water pressure, earth pressure, weight of equipment, etc. Ideally, design of reinforced concrete structures by computer should be done in an interactive mode on remote terminals. This would give the designer the most flexibility in combining the many shapes to fit the design criteria. Better yet, would be to allow the designer to do his work on graphical display devices.

Civil engineers at the Carnegie-Mellon University have recognized a need for a problem-oriented language in the field of water resources technology. A pilot language, HYDRO, has been made operational for a segment of this field. A language, such as this, should be developed more extensively and made available to the engineering community.

Equipment selection and cost estimating

Today the designer of treatment plants makes a selection of possible equipment to suit his given criteria from a handful of possibilities that he is familiar with, possibly neglecting other equipment more suited to his conditions. A data bank of performance characteristics and other information pertinent to the full range of equipment used in treatment plants and pertaining to a large number of manufacturers could assist these designers. This information could also be used in determining projected operating costs under a variety of conditions.

Some progress has been made in assisting the engineer in making construction cost estimates, but much more could be done. One problem is the quantifying of the numerous items that go into the construction of these facilities. Another major problem is that of making reasonable estimates of unit costs for each of these items. With the costs varying from contractor to contractor, with the fluctuations in the cost of labor, with the general inflationary trend, and all of these varying in the different sections of the country, a consultant has a difficult time in keeping track of the latest and most reasonable unit prices. This is especially true when one considers that a consultant rarely has the same type of job in the same section of the country with a frequency such that the prices hold true from one job to another. A national data base could be established and be updated on a regular basis with latest prices as bid by contractors on various types of jobs throughout

the country. This data base could be accessed by consultants through the use of remote terminals.

Drafting systems

A number of papers have already been written on establishing automated drafting systems and some of the leading large companies are now working with the first of such systems. Suffice it to say here that such systems are also needed in the fields of civil and sanitary engineering. However, due to the size of the companies involved, these systems must be available at much lower prices, and in a time-sharing mode, before these consultants can make use of them.

Making application programs useful

Simply automating these problems for the engineer is not enough. In the development of programs the usability factor must always be kept in mind. A program is almost worthless if the user finds the input requirements too cumbersome or the output strange and illogically presented. Under such conditions the engineer will probably, and has been known to, revert to his familiar manual methods.

In any printed output which the engineer must utilize in his design work, the results *must* be in terms that he will be able to readily use and understand. This may seem elemental, but non-computer-oriented friends have told me that more than a few times, when people outside the profession (such as programmers who are more oriented to other disciplines, mathematicians, etc.) have produced analytical programs, the output is presented in an academic or even unfamiliar manner. Often the output does not reflect the way things are normally done within the profession, and it causes problems in understanding, together with creating a distrust in the program as well as the programmer.

Similarly, I know that when a mathematician was outlining the requirements of a study of the Ohio River, he insisted that data concerning certain flow characteristics of the river had to exist or the program could not be written. In actuality, records which had been kept did not contain such data because of the impracticality of obtaining them. The practicing engineer had to either do without it, or analyze around it. The mathematician was so involved with theory that he could not accept the impure practicalities of the situation.

In other instances, certain proprietary programs have been obtained that output only part of the data required for fully understanding the printed results.

What I am trying to say is this: learn to speak the language and fully understand the problems and

idiosyncrasies of the job to be done. Simply automating the procedure is not enough. This has been painfully clear and exasperating to many housewives when they receive their monthly billing from many department stores. All they see on the bill is an amount of a purchase and a department number. This information may be fully adequate for the store, but the housewife (and many of us husbands, for that matter) could care less what the department number is. What is needed is the date and a recognizable name of the purchased item, as well as the price. While the programmer or systems analyst planned the output for the accounting department, he did not plan sufficiently for all the expected end-users of that output. Again, simply automating the procedure is not enough. A presentation meaningful to the end-user must always be kept in mind. Merely understanding the problem from a data processing standpoint is not sufficient; we must strive to understand more fully the language and the ramifications of the problem at hand.

I have mentioned the difficulty that is generated when inadequate or insufficient data is printed out or displayed for the end-user. In many technical problems there is much data that could be included on the output forms and often there is not enough space to show it all conveniently. The programmer may select the data he thinks is most important for the output. However, an engineer may desire different data or a different output format. Other engineers may desire still other data or formats. A means should be developed to enable a user of a program to tailor the data and formats to his own requirements without having to reprogram the output routines. This is especially true where the user is not a programmer, or does not own the source code of a leased program.

SUMMARY

I have discussed a number of specific areas, along with some more general areas in which programming work needs to be done to help get some of our pollution problems solved. All of them have one overriding premise, however. And that is that the use of computers be made more practical for the non-computer-oriented professional engineer, and that programs be available at a reasonable cost. This means that terminal-oriented time-sharing systems must be made easier to use by the layman, and that libraries of these specialized programs be made available, probably on a royalty-per-use basis. Also, a carefully planned method of making such programs available, as well as the means of obtaining them, should be familiar to all concerned.

With some \$400 million worth of design effort (\$16

billion worth of water and sewerage treatment projects) already backlogged and much more to come, the need is now here for concerned people in the programming profession to lend assistance to the battle against water pollution in this country. A great many of us have come to programming from other previous vocations, and many of these vocations can have a bearing on some facet of the problems facing the sanitary engineer. It behooves us all to give serious thought to how our various backgrounds can be applied to these problems. By joining forces, we may be able to hasten the day when we can hear that the water pollution problems are under control.

ACKNOWLEDGMENTS

The author would like to express his gratitude to the following persons for their valuable ideas and suggestions, many of which have been incorporated in this paper: Mr. Frank Perkins, Professor of Civil Engineering at the Massachusetts Institute of Technology; Mr. Richard Foerster, and Mr. Paul E. Langdon, Jr., both of Greeley and Hansen, Engineers, in Chicago, Illinois; and Mr. Rodney Dabe, of Consoer, Townsend & Associates, Consulting Engineers in Chicago, Illinois.

BIBLIOGRAPHY

- 1 *ENR's 93rd annual report and forecast*
Engineering News-Record Vol 180 no 4 pp 60 & 74
Jan 25 19 8
- 2 *ENR's 94th annual report and forecast*
Engineering News-Record Vol 182 no 4 p 64 Jan 23 1969
- 3 *Construction scoreboard*
Engineering News-Record Vol 182 no 4 p 146 Jan 23 1969
- 4 *95th annual report and forecast*
Engineering News-Record Vol 184 no 4 p 56 Jan 22, 1970
- 5 *Construction scoreboard*
Engineering News-Record Vol 184 no 4 p 130 Jan 22 1970
- 6 *96th annual report and forecast*
Engineering News-Record Vol 186 no 3 p 23 Jan 21 1971
- 7 *Construction scoreboard*
Engineering News-Record Vol 186 no 3 p 91 Jan 21 1971
- 8 *Display helps fight pollution*
Battelle Memorial Institute
Electro-Technology p 18 June 1969
- 9 *New York uses computer to monitor pollution*
Datamation pp 63-64 Sept 1970
- 10 E S MUSKIE US Senate
Computers environmental planning, and the quality of life
Proceedings of IBM Scientific Computing Symposium on
Water and Air Resource Management May 1968
- 11 V L HOBerecht
Computer aided design and drafting
IBM Technical Report 21.244 March 1967
- 12 D P LOUCKS C S REVELLE W R LYNN
Linear programming models for water pollution
Management Science p B166-B181 Dec 1967

- 13 A GOTTLIEB
The computer and the job undone
Computers and Automation pp 16-23 Nov 1970
- 14 *Integrated civil engineering system (ICES) for programming*
Computers and Automation pp 10-11 April 1968
- 15 G BUGLIARELLO
*Programming needs in the water resources field and the
role of a problem oriented language (Hydro)*
IBM Scientific Computing Symposium on Environmental
Sciences pp 165-184 Sept 1967
- 16 P R DECICCO H F SOEHNEN J TAKAGI
Use of computers in design of sanitary sewer systems
Journal of Water Pollution Control Federation
Vol 40 no 2 part 1 pp 269-284 Feb 1968
- 17 Proceedings of IBM Scientific Computing Symposium on
Water and Air Resource Management 392 pp May 1968
- 18 R SMITH
*Preliminary design and simulation of conventional
wastewater renovation systems using the digital computer*
U S Environmental Protection Agency Water Quality
Office 3-1968 WP-20-9

Application of a large scale nonlinear programming problem to pollution control*

by GLEN W. GRAVES

University of California
Los Angeles, California

and

DAVID E. PINGRY and ANDREW WHINSTON

Purdue University
Lafayette, Indiana

INTRODUCTION

In recent years it has been recognized by several observers that the techniques of mathematical programming can be used to select a least-cost solution to the problem of river quality maintenance. In general these models have used the solution technique of linear programming and considered one treatment alternative, such as on site treatment or by-pass piping. Examples of these models can be seen in Deininger,¹ Louchs, ReVelle and Lynn² and Graves, Hatfield and Whinston.³ The use of these techniques has allowed, in a theoretical context, large reductions in total treatment costs in a river basin.

The procedure used in constructing river basin models has been to divide the river into small sections and place constraints on the water quality at the end of these sections. In all cases the water quality criteria used is the level of dissolved oxygen concentration. The level of dissolved oxygen at the end of the river sections is calculated using the Streeter-Phelps equations or some later variation. Cost functions are then estimated and a mathematical programming problem of the following form is solved:

Minimize: The total cost of pollution abatement structures
Subject to: Water quality in each section of river better than some given set of quality goals

* This research has been sponsored by the Office of Water Resources Research under Contract 14-31-0001-3080. The authors are responsible for all possible errors.

It is the purpose of this paper to present a model of a river basin of the form suggested. We will consider simultaneously the following treatment methods:

- (1) Flow augmentation
- (2) By-pass piping
- (3) Treatment plants (regional and at polluter)

We also show that linear constraints, consistent with the linear programming technique, are not appropriate when these treatment alternatives are considered. Therefore, a nonlinear programming algorithm must be used. Some details of this algorithm are discussed and the model is applied to the White River Basin in Indiana.

WATER QUALITY MEASURES

Water quality can be measured in a variety of ways. The appropriate parameter or set of parameters measured depends on the intended use of the water.

Traditionally, water quality in rivers has been measured by looking at the level of dissolved oxygen concentration. This parameter has been used because of its direct relationship with the type and quantity of living organisms in a body of water. If the level of dissolved oxygen should drop to zero the river is said to be "septic." In this condition only anaerobic organisms can exist. These types of organisms rely on oxygen which is in compounds rather than free oxygen which, in the case of a septic river, is not available. In the process of freeing the oxygen from compounds the

anaerobic organisms produce by-products which often cause the obnoxious odors and colors which appear in polluted waters.

When effluent, such as common sewage, is dumped into a river it creates additional demand for oxygen over and above the demands of the existing living organisms. If this additional load is moderate then the river can recover using the oxygen entering at the surface. If the additional load is low enough it is possible that the reduction in the dissolved oxygen concentration level will be acceptable. However, if the additional load is high the river may become septic before it can recover, and if additional heavy loads are dumped into the river as it proceeds downstream, it may never recover, and in fact remain septic for its entire length.

The oxygen required for the oxidation of organic matter is called biological oxygen demand or (BOD). The dissolved oxygen concentration level is often measured relative to the dissolved oxygen saturation level of the water and is called the dissolved oxygen deficit (DOD).

The level of the dissolved oxygen concentration in a body of water is a function of the amount of oxygen being absorbed at the surface from the air and the amount being consumed in the water by the biochemical oxidation of organic material. Since, both the consumption of the oxygen by organic material and the absorption at the surface are not instantaneous reactions, a sophisticated method of predicting the effect of an organic material on the level of DOD after a given period of time is necessary. The first successful attempt to mathematically describe this relationship was by Streeter and Phelps.⁴ Their work has only been slightly modified to this date. The model used was a set of differential equations, given in (1) and (2).

Assume:

$$db_k/dt = K_{1k}b_k \quad (1)$$

$$dd_k/dt = K_{1k}b_k - K_{2k}d_k \quad (2)$$

The terms used in (1) and (2) are defined as follows:

- t = time of reaction (days)
- K_{1k} = rate of oxidation reaction (days⁻¹)
(deoxygenation rate)
- K_{2k} = rate of absorption of oxygen (days⁻¹)
(reaeration rate)
- b_k = BOD concentration (mg/ ℓ)
- d_k = DOD concentration (mg/ ℓ)

Equations (1) and (2) are integrated to yield equations (3) and (4).

$$b_k = b_k^B C_{1k} \quad (3)$$

$$d_k = K_k b_k^B [C_{1k} - C_{2k}] + d_k^B C_{2k} \quad (4)$$

The terms K_k , C_{1k} and C_{2k} are defined in (5)-(7).

$$K_k = K_{1k} / (K_{2k} - K_{1k}) \quad (5)$$

$$C_{1k} = \exp(-K_{1k}t) \quad (6)$$

$$C_{2k} = \exp(-K_{2k}t) \quad (7)$$

b_k^B and d_k^B are the values of BOD and DOD when $t=0$.

This now enables one to predict the value of DOD at some point in time after the introduction of an extra load of BOD. If the body of water with which we are concerned is a river, and if over a small segment of a river the velocity of flow is assumed constant, then the length of time that the reaeration and reoxygenation reactions take place in that segment is a linear transformation of the length of the segment. This is expressed in Equation (8).

$$t = (1/V_k)x_k \quad (8)$$

X_k is the length of the river segment and V_k is the velocity of flow assumed in that section.

Using the assumption of constant velocity, the values of b_k and d_k can be interpreted as the values of BOD and DOD at the end of river segment k . In equations (3) and (4) they are written as a function of the initial values of BOD and DOD, b_k^B and d_k^B , given values of the parameters K_{1k} , K_{2k} , V_k and X_k for that section.

The use of the dissolved oxygen theory in the context of river basin programming models has been influenced by the desire of the researchers to maintain a set of linear quality constraints. This linearity has been maintained by two procedures. The first is to assume the parameters K_{1k} , K_{2k} and V_k constant for a given segment of the river.²

A second procedure used to maintain linearity is the method first proposed by Thoman.⁵ This approach views the river as a black box where effluent is dumped and dissolved oxygen levels are changed in some manner unknown to the researcher. A matrix of so called transfer coefficients is generated in which each element, a_{ij} , is the marginal effect of a change in the BOD level in section j on the DOD in section i . This concept is used in the programming models constructed by Graves, Hatfield and Whinston³ and Schaumburg.⁶

Both of these linear representations are fairly accurate as long as the flow is not allowed to vary to a significant degree. The reason for this restriction is that it has been found that the level of the flow affects the reaeration rate and the velocity of the flow in a particular river segment. If this is indeed the case, and the level of effluent flow is large relative to the river flow, then the effect of the flow on the reaeration coefficient is not negligible as assumed in most programming models. If we assume that the values of K_{2k} and

V_k are a function of flow, then from Equations (3) and (4) we see that the values of d_k and b_k can no longer be represented as a linear combination of b_k^B and d_k^B . This in turn implies that any quality constraint formed for use in a programming model will be nonlinear in nature.

We also note that if flow augmentation is to be used as a treatment alternative, the level of flow will not only affect the BOD and DOD concentration by dilution, but also by altering the value of K_{2k} . This points to the necessity of having a programming algorithm which can properly handle nonlinear constraints.

Equations (9) and (10) give the relationship between flow and velocity, and between flow and the reaeration rate assumed in our model.

$$K_{2k} = g_k F_k^{h_k} \tag{9}$$

$$V_k = y_k F_k^{z_k} \tag{10}$$

The parameters g_k , h_k , y_k and z_k must be estimated for each river section for a particular application of the Streeter-Phelps equations.

It is important to note that the selection of dissolved oxygen as the measure of water quality for the application of our model does not imply that this is the only standard which could easily be used.

RIVER BASIN MODEL

We will now formulate a simulation model of a river basin which can be used in combination with the quality model discussed in the previous section to predict the level of DOD at a finite number of points along a river.

In order to construct the model the river is divided into n sections. A new section begins where one of the following occurs:

1. Effluent flow enters the river.
2. Incremental flow enters the river. (Ground water, tributary flow, etc.)
3. The flow in the main channel is augmented or diverted.
4. The parameters describing the particular river change.

Assume that there are s polluters and m treatment plants in the river basin. Each polluter is able to pipe effluent either directly into any segment of the river or to any of the m treatment plants. Each treatment plant can in turn pipe to any of the n sections.

The system of pipes described allows for the possibility of by-pass piping and regional treatment plants. The importance of by-pass piping as a treatment alternative was demonstrated in the study of the Delaware River done by Graves, Hatfield and Whinston.³

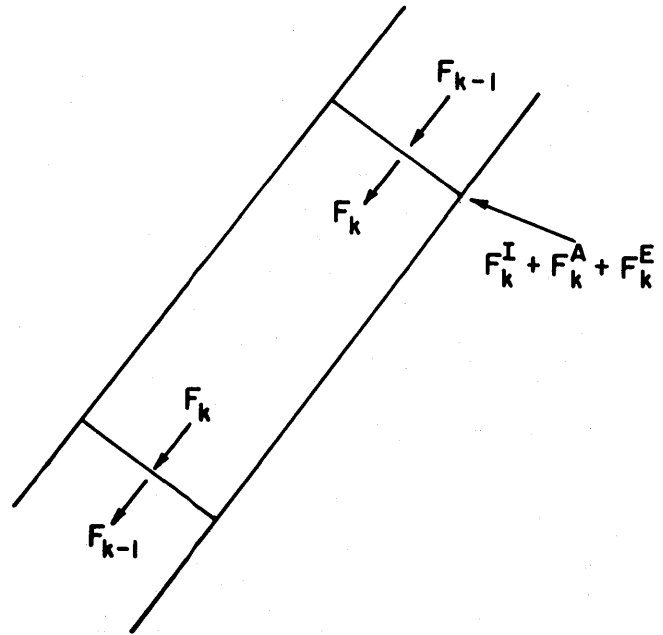


Fig. 1

The purpose of this method of treatment is to transport waste from densely populated or industrialized regions to low use areas to take advantage of the natural treatment capabilities of the river. Regional plants would be constructed to combine the wastes of two or more polluters to take advantage of economies of scale which exist in the production of treated wastes.

The water quality model, as discussed above, can now be used in the form of Equations (3) and (4) to calculate the value of DOD at the end of each of the n river segments.

In order to use Equations (3) and (4) we must know the values of BOD and DOD at the head of each section. The concentrations, b_k^B and d_k^B , are a weighted average of the concentrations of the flow of section $k-1$ and all of the effluent, incremental and augmentation flows entering section k . These relationships are expressed in Equations (11) and (12).

$$b_k^B = [b_{k-1}F_{k-1} + b_k^A F_k^A + b_k^E F_k^E + b_k^I F_k^I] / F_k \tag{11}$$

$$d_k^B = [d_{k-1}F_{k-1} + d_k^A F_k^A + d_k^E F_k^E + d_k^I F_k^I] / F_k \tag{12}$$

The terms F_{k-1} , F_k^A , F_k^E and F_k^I represent, respectively, the flow from section $k-1$, augmentation flow in section k , effluent flow in section k and incremental flow in section k . The b terms represent the associated BOD concentrations and the d terms the associated DOD concentrations. See Figure 1 for an illustration of a typical section.

The effluent flow entering each river section will be the sum of the flows coming from polluters directly and from treatment plants. The BOD and DOD concentrations of these flows will be the weighted average of the concentrations of all the flows. In turn, the BOD and DOD concentrations of the flows from treatment plants will be a weighted average of all the flows entering that plant times the treatment levels.

Given the values of the BOD and DOD concentrations at the polluters, the percentage of BOD removal at each treatment plant and the values of the various pipe flows the values of b_k^B and d_k^B can be calculated for any k . If $k > 1$ the Streeter-Phelps equations must be applied sequentially to all sections i , for which $i < k$ to obtain b_{k-1} , d_{k-1} and F_{k-1} .

The values of the incremental and augmentation flows and their associated concentrations must also be known. The value of F_k^I will be some fixed constant which will account for groundwater, small tributaries, diversions for water supplies or industrial use or any other fixed inflow or outflow. The augmentation flow will be some functions of the size of the reservoir which provides the necessary storage. In our model the augmentation flow will be a variable flow available at the sites of potential reservoirs in the basin. The cost associated with the different levels of flow will depend on the particular site.

The river simulation model developed above will now be integrated into a nonlinear programming model which will provide, by means of the objective function, a decision-making criteria to select least-cost treatment abatement program for a given set of quality goals. The simulation model is used, in the context of the programming problem, to evaluate the water quality constraints. These constraints will be of the following form:

$$d_k - \bar{d}_k - \Delta d_k \leq 0 \quad k = 1, n \quad (13)$$

The value of d_k is the DOD concentration at the end of reach k and \bar{d}_k is the DOD concentration before any structures other than existing structures are constructed. The values of Δd_k contain the information concerning the water quality goals of the society or governmental unit who must decide which of the infinitely possible water quality standards is the "right" one. For example, the values of Δd_k could be set so that 5 mg/l of dissolved oxygen must be maintained at every section on the river. This is known as a uniform river standard. Perhaps the river could be divided into zones or regions, each having a different water quality standard. An example of this, would be to maintain higher water quality around cities for recreational use and allow the river to deteriorate to a lower level of

dissolved oxygen in rural low use areas. The pattern of quality demanded by any of these alternate schemes can be reflected in the values chosen for Δd_k . Many examples of these types of schemes can be seen in the papers by Schaumburg⁶ and Graves, Hatfield and Whinston.⁷

A programming problem of the form discussed above is formalized:

$$\text{Minimize: } TC = C^P + C^{TP} + C^R$$

where,

C^P = Total cost of all pipelines constructed in river basin.

C^{TP} = Total cost of all treatment plants constructed in river basin.

C^R = Total cost of all reservoirs constructed in river basin.

Subject to: $d_k - \bar{d}_k \leq \Delta d_k \quad k = 1, n$

In order to keep the mathematical representation consistent with the physical reality two more sets of constraints must be added to the programming model. The first of these is to keep the total flow leaving a polluter in pipes to treatment plants and river sections equal to the total flow of effluent available at that polluter. Another set of necessary constraints maintains consistent flow entering and leaving a particular treatment plant.

The first n quality constraints are nonlinear inequality constraints. The flow conservation constraints are linear and are equality constraints. The objective function, which will be detailed in a later section, is also nonlinear. This type of problem requires a general nonlinear algorithm to obtain a solution. The algorithm required will be discussed in the next section.

NONLINEAR ALGORITHM

The algorithm employed for solving the nonlinear programming problem of this paper is a general purpose algorithm which solves problems of the form:

$$\text{Subject to: } g^i(y) \leq 0 \quad i = 1, m-1 \quad (14)$$

Minimize: $g^m(y)$

where y is a vector in E^n and $g^i(y)$, $i = 1, m$, are continuous functions with continuous partial derivatives defined on some open set. The vector y is assumed to be bounded from above and below.

The method to be discussed here was originally described by Graves.⁸ The method can also be used as a

second order procedure as presented by Graves and Whinston.⁹ This paper will be limited to the discussion of the algorithm as it was used to solve the large scale water pollution problem described in the previous sections. A theoretical proof of convergence will be presented elsewhere.

The algorithm to be described is stepwise in nature. Starting with some point y^j in the domain of the functions, a direction Δy^j and a scalar k are determined and a new point y^{j+1} is calculated.

$$y^{j+1} = y^j + k\Delta y^j \tag{15}$$

The vector Δy^j is also a vector in E^n .

The object of making the step to y^{j+1} is to either reduce the value of the objective function, if y^j is a feasible solution to the nonlinear programming problem, or obtain a "more feasible" solution to the nonlinear problem, if y^j is an infeasible solution. The phrase "more feasible" is interpreted in terms of the algorithm to mean "reduce the value of SUPG", where SUPG is defined to be:

$$\text{SUPG} = \text{Max}\{g^i(y^j) \mid g^i(y^j) \geq 0\}$$

or alternatively stated:

$$\text{SUPG} = g^w(y^j)$$

where w is the index of the most infeasible constraint. If y^j is a feasible solution to (14), then $\text{SUPG} = 0$.

The completion of the determination of Δy^j and k , and the calculation of y^{j+1} will complete what will be known in the paper as the $j+1$ th nonlinear iteration. Each nonlinear iteration will consist of several local linear programming problems to determine Δy^j . The solution of each one of these linear programming problems will complete what will be known as a linear iteration.

For the purpose of our exposition the nonlinear iteration will be divided into two major parts. The first is the determination of Δy^j as a solution to a parametric linear programming problem. The second is the determination of k .

Since we have assumed that the functions $g^i(y^j)$, $i = 1, m$, are continuous, and have continuous partial derivatives on some open set D , the following approximation theorem can be used:

Let $\nabla g^i(y^j)$ be the gradient of $g^i(y^j)$ at the point y^j , where y^j is a member of a closed bounded subset E of the open set D .

Then,

$$g^i(y^j + \Delta y^j) = g^i(y^j) + \nabla g^i(y^j)^T \Delta y^j + R^i(\Delta y^j)$$

where

$$\lim_{\Delta y^j \rightarrow 0} R^i(\Delta y^j) / |\Delta y^j| = 0$$

uniformly for $y^j \in E$.

The direction of improvement for nonlinear iteration $j+1$ is obtained from the function above by estimating the term $R^i(\Delta y^j)$, and solving the associated local linear programming problem.

Subject to:

$$\nabla g^i(y^j)^T \Delta y^j \leq -g^i(y^j) - \bar{k}r^{ij} \tag{16}$$

Minimize:

$$\nabla g^m(y^j)^T \Delta y^j$$

The r^{ij} term is the estimated error for the i th equation during the $j+1$ th nonlinear iteration. The value of r^{ij} is determined during the nonlinear iteration using Equation (17)

$$r^{ij} = g^i(y^{j-1} + \Delta y^{j-1}) - g^i(y^{j-1}) - \nabla g^i(y^{j-1})^T \Delta y^{j-1} \tag{17}$$

where k is implicitly assumed to be one. The absolute value of r^{ij} is used for the linear programming problem. This is required for the purpose of the convergence theorem.

The parameter \bar{k} will be adjusted in the course of the nonlinear iteration. The value of \bar{k} is greater than, or equal to zero, and is estimated from the length of the previous step. The role of \bar{k} in the linear and nonlinear problem will be discussed in detail later in this section, as will the estimating procedures used to obtain \bar{k} .

The linear programming problem (16) can be treated as a parametric programming problem with \bar{k} as the modifying parameter and can be written in tableau form as illustrated in Tableau I.

Tableau I

	Δy^j	$(VBV)_p$	$(BV)_p$
$-x_1$	$\nabla g^1(y^j)^T$	$-g^1(y^j) - \bar{k}r^{1j}$	z_1
\vdots	\vdots	\vdots	\vdots
$-x_{m-1}$	$\nabla g^{m-1}(y^j)^T$	$-g^{m-1}(y^j) - \bar{k}r^{m-1,j}$	z_{m-1}
$(VBV)_d$	$\nabla g^m(y^j)^T$	$-g^m(y^j) - \bar{k}r^{mj}$	$-z_p$
$(BV)_d$	$-v$	$-z_d$	

The vectors Δy^j and v are $n \times l$. The members of vector Δy^j are the primal variables, and the members of vector v are the slack variables of the dual problem. The vector of the dual variables is

$$x^T = [x_1, \dots, x_{m-1}]$$

and the vector of the primal slack variables is

$$z^T = [z_1, \dots, z_{m-1}]$$

Φ_p is the value of the primal objective function and Φ_d is the value of the dual objective function. The labels $(VBV)_p$ and $(VBV)_d$ are respectively the values of the current basic primal variables and the basic dual variables. The labels $(BV)_p$ and $(BV)_d$ are the basic variables associated with the given values.

From the duality theorem of linear programming there are three possible termination conditions to the local linear programming problem.

- (A) There exists an optimal feasible solution to the primal and dual problems.
- (B) The constraints for the primal problem are infeasible and the dual problem is unbounded or the constraints of the dual problem are inconsistent.
- (C) The primal problem is unbounded and the dual problem is infeasible.

The initial solution in the domain of the functions $g^i(y^j)$, $i = 1, m$, is not required to be a feasible solution to the nonlinear problem stated in (14). As was discussed above, if y^j is not a feasible solution to (14), then $SUPG > 0$. If y^j is a feasible solution to (14), then $SUPG = 0$. The goal of each nonlinear iteration is either to reduce the value of the objective function $g^m(y^j)$, or move closer to feasibility, which is interpreted to mean reduce the value of $SUPG$.

The case of nonlinear infeasibility will usually imply that the local linear problem (16) will be infeasible for any Δy^j in the ϵ region around y^j . In this case $\nabla g(y^j)^T \Delta y^j$ is chosen as the objective function and a linear programming problem such as (18) will be constructed.

Subject to:

$$\nabla g^p(y^j)^T \Delta y^j \leq -g^p(y^j) - \bar{k} r^{pj} \quad p \in H$$

Minimize:

$$\nabla g^I(y^j)^T \Delta y^j \quad (18)$$

where

$$H = \{i \mid \nabla g^i(y^j)^T \Delta y^j \leq -g^i(y^j) - \bar{k} r^{ij} \text{ for some } \Delta y^j\}$$

Since this problem is consistent and bounded it can only terminate in condition (A). However, it is still possible that the entire linear problem (42) is infeasible,

or in other words terminates in condition (B). In this case the nonlinear problem will also be infeasible at the new point, y^{j+1} , ignoring errors. Mathematically this would mean that

$$\Delta g^i(y^j)^T \Delta y^j > -g^i(y^j) - \bar{k} r^{ij} \quad \text{for some } i, i = 1, m-1. \quad (19)$$

However, if a gain has been made in SUPG, then the algorithm proceeds through the nonlinear iteration with the determination of \bar{k} .

Of course if the gain is large enough feasibility may be reached and the local linear problem would terminate in condition (A).

If the local linear programming problem (16) terminates with condition (B) holding and no gain has been made in SUPG, then it is assumed that the nonlinear problem is inconsistent and the algorithm terminates unless \bar{k} can be adjusted as will be discussed later.

The other possible termination condition (A) implies that a feasible solution to the entire linear problem (16) has been obtained, and ignoring errors, y^{j+1} is a feasible solution to the nonlinear problem. The algorithm at this point will check for a gain in $g^m(y)$. If at the new $y^{j+1} = y^j + \Delta y^j$, there is no gain in $g^m(y)$, then we assume that the local minimum has been reached. If there is a gain in $g^m(y)$, then another nonlinear step is taken.

At this point it is necessary to explain in some detail the role that the parameter \bar{k} plays in the final determination of Δy^j . In order to see the role \bar{k} plays more clearly, it is necessary to write mathematical expressions for the statements, "gain in $g^m(y)$," and "gain in SUPG." If the local linear problem terminates in condition (A), then

$$\nabla g^m(y^j)^T \Delta y^j = \sum_{i=1}^{m-1} (-x_i) g^i(y^j) + \bar{k} \sum_{i=1}^{m-1} (-x_i) r^{ij} \quad (20)$$

In order for a gain to be made in the nonlinear objective function, the following inequality must hold:

$$\nabla g^m(y^j)^T \Delta y^j < -\epsilon \quad (21)$$

Using Equations (20) and (21), the condition for a gain in $g^m(y^j)$ is

$$\sum_{i=1}^{m-1} (-x_i) g^i(y^j) + \bar{k} \sum_{i=1}^{m-1} (-x_i) r^{ij} < -\epsilon. \quad (22)$$

At this point, the dual variables are less than, or equal to zero. The r^{ij} are assumed greater than or equal to zero, and since the nonlinear problem is feasible $g^i(y^j) > 0$. This information implies that:

$$\sum_{i=1}^{m-1} (-x_i) g^i(y^j) < 0 \quad (23)$$

and

$$\bar{k} \sum_{i=1}^{m-1} (-x_i) r^{ij} > 0 \quad (24)$$

From (22) and (24), it is clear that as \bar{k} approaches zero, the gain in $g^m(y^j)$ would be greater. Therefore, if the linear problem terminates in condition (A) and there is no gain in $g^m(y)$, then \bar{k} can be adjusted downwards, which effectively is relaxing the linear constraints. As \bar{k} goes to zero, condition (22) becomes

$$\sum_{i=1}^{m-1} (-x_i) g^i(y^j) < -\epsilon \quad (25)$$

The same sort of condition can be derived in the case when the linear programming problem terminates in condition (B). Condition (26) is for a gain in SUPG.

$$\sum_p (-x_p) g^p(y^j) < -\epsilon \quad p \in H \quad (26)$$

Using these results, the criteria that the algorithm uses for nonlinear optimality and nonlinear infeasibility can be written as follows:

Optimality:

If \bar{k} is adjusted as low as possible, the linear program terminates in condition (A) and

$$\sum_{i=1}^{m-1} (x_i) g^i(y^j) \geq \epsilon$$

then y^j is assumed to be the optimal solution to the nonlinear problem.

Infeasibility:

If \bar{k} is adjusted as low as possible, the linear program terminates in condition (B), and

$$\sum_p (-x_p) g^p(y^j) \geq -\epsilon \quad p \in H$$

then the constraints of the nonlinear problem are assumed to be inconsistent.

Before the steps of the actual program are discussed one additional feature of this algorithm must be mentioned. It is possible to divide the n variables in the nonlinear problem into IPRN priority classes. For example, assume that IPRN = 2. This implies that every variable is either in priority class one or two. All of the variables in priority class one would be used to try and obtain a gain in SUPG or $g^m(y)$. The second priority class variables would not be considered unless no gain could be made using the priority one variables with \bar{k} adjusted to zero. The number of priority classes is unlimited.

Using the criteria described above for optimality and infeasibility, the steps actually taken in the computer program are described in sequence:

1. The variables, Δy^j , which are currently not in the basis of the linear programming problem, are scanned for possible entry. All of the variables will be out of the basis at the outset of each nonlinear iteration. The scanning is accomplished by updating the element associated with the current linear objective function. If priority classes are used, then only those variables which have a priority level less than or equal to the current level, IPRC, are checked.
2. The variable associated with the updated element of highest absolute value is selected to enter the linear tableau. This criteria is used because this variable locally affects the objective function more than the other variables.
3. The element with the largest absolute value is tested to see if it is significantly different from zero. If it is, the algorithm proceeds to step 4 and the solution of the linear programming problem. If not, it is assumed that the addition of the variable associated with the largest element would not affect the objective function significantly, since the appropriate coefficient is so small. In this case, \bar{k} is adjusted downwards, or if $\bar{k} = 0$, the number of priority classes considered is expanded. If the current priority class is the last one available, then the algorithm will terminate. The termination will mean one of two things; the nonlinear problem is infeasible since no gain can be made in SUPG = $g^m(y) > 0$, or the local minimum to the nonlinear problem has been attained and no gain can be made in $g^m(y)$.
4. After selecting the variable column to be added to the linear programming problem, the linear programming tableau is augmented by the new column and if a gain was made in the linear objective function on the previous linear iteration, the columns associated with the variables rejected from the basis are removed from the linear tableau.
5. The linear programming algorithm now takes over and solves the local problem set up in the previous steps. The linear programming problem will terminate in one of the three terminal conditions discussed above. If terminal condition (A) is reached, linear feasibility, then the objective function $g^m(y)$ is tested to see if a gain greater than some tolerance level was made. If terminal condition (B) is attained, then the algorithm tests for a sufficient gain in SUPG.

If either of these gains are successfully made, the algorithm leaves the linear programming subproblem. If the gains are not made, then the algorithm returns to selecting variables to enter the linear tableau. If terminal condition (C) is reached, the algorithm again returns to selecting variables in order to bound the primal problem.

The first step after the successful conclusion of the local linear problem is to select the "best" value of \bar{k} . This calculation is called the post-optimal adjustment and proceeds in two different manners, depending on the value of SUPG.

If the value of SUPG is zero, or we have nonlinear feasibility, then the value of the objective function is written as a function of \bar{k} , and this function is solved for the value of \bar{k} , which will minimize $g^m(y)$.

In the case where SUPG > 0, or we have nonlinear infeasibility, \bar{k} is approximated by choosing a trial value such that G is zero, where G is defined to be

$$G = (\Delta y^{j-1})^T (\Delta y^{j-1}) - (\Delta y^j)^T (\Delta y^j) \quad (27)$$

In either case, SUPG = 0, or SUPG > 0, the value of \bar{k} must be tested to see that it does not violate bounds which are implied by the bounds on y . If the value of \bar{k} determined in the post optimal adjustment violates the greatest lower bound on \bar{k} , or results, in the case of SUPG = 0, in no gain in $g^m(y)$, the value of \bar{k} is adjusted downwards. The control of the problem is then passed back to the linear programming part of the algorithm.

It is at this point the new estimates of the error terms r^{ij} are calculated. This is done by evaluating the functions $g^i(y^j + \Delta y^j)$, $i = 1, m-1$, and using Equation (17). The new values of r^{ij} are used for the rest of the $j+1$ -th nonlinear iteration and the absolute values are used for the local linear problems in nonlinear iteration $j+2$.

The next step in the algorithm is to calculate the range of values for k which will maintain a feasible solution or give the best gain in SUPG. After a range of values is determined, the optimal value of k is chosen from the range determined.

The range is calculated by using the following equation:

$$g^i(y) = g^i(y^j) + g^i(y^j) \Delta y^j k + k^2 r^{ij} \leq D(\text{SUPG}) \quad (28)$$

If SUPG = 0 then Equation (28) just says that the new values of $g^i(y)$, $i = 1, m-1$, must be feasible. If SUPG > 0, then D is initially set equal to zero. The quadratic functions in k are then solved.

$$g^i(y^j) - D \text{ SUPG} + g^i(y^j) \Delta y^j k + k^2 r^{ij} = 0, \\ i = 1, m-1 \quad (29)$$

This will give the value or values of k where the constraints $g^i(y)$ will go infeasible. If the lower bound on k is greater than the upper bound, then the value of D is increased and the quadratic problem is again solved. If as D approaches one, the upper bound continues to be lower than the lower bound, then we say that the interval determination failed and no k can be found which will improve SUPG. In this case the algorithm terminates.

If SUPG = 0, the algorithm determines the interval which will maintain the feasibility of y . After the interval is determined, the best value of k is found by evaluating the function $g^m(y^j + k \Delta y^j)$ for different k 's in the range given. The gain in the objective function is checked to see if it exceeds some preset tolerance level. If not, then it is assumed that we are within the length of that tolerance level of a local optimal solution and the algorithm terminates.

If a gain in SUPG is made, or a reduction in $g^m(y)$, the algorithm takes another nonlinear iteration. This procedure is repeated until either a local minimum or infeasibility is encountered.

APPLICATION OF RIVER BASIN MODEL

The model as proposed has been applied to the West Fork White River in Indiana. The West Fork White River has its source near the Indiana-Ohio border. The general direction of flow is southwesterly for 371 miles through the State of Indiana. The major city on the river is Indianapolis, which is 234 miles from the mouth. Two minor cities, Anderson and Muncie, are upstream from Indianapolis. The concentration of population and industry around these three cities causes the major portion of the pollution problem in the West Fork.

For the purpose of this paper we have chosen a length of the West Fork White River, which runs from the headwaters above Muncie to just south of Indianapolis. The portion described is 133.2 miles long. It has been divided into 46 sections based on information about polluters, incremental flow, and river parameters. The sections range in length from .1 mile to 12.2 miles, with most sections in the 2.0 to 5.0 range.

In the implementation of the model, some alterations were made which were not explicit in the exposition of the model in the previous sections of the paper.

The first of these alterations is that the number of river sections, number of polluters and the number of treatment plants are assumed to be equal, and a potential polluter and treatment plant are located at the beginning of each river section. In terms of the model

presented above, this assumption can be written as $n=m=s$. In any given problem the flow from some of the polluters will be zero (i.e., no polluter exists currently at that point on the river). The treatment plants in those sections are potential regional treatment plants. In the section where polluters do exist, the treatment plants are the on-site treatment plants for the associated polluter, and can also act as potential regional treatment plant sites. Regional treatment plants can be located anywhere along the river by creating a new river section. It is not at all necessary to make this alteration, but it does allow for the easy addition of polluters at a later date, and permits easy calculation of the incremental cost of these additional polluters.

The second of these alterations makes it possible to reduce the number of variables in the system. The desirability of this can be seen by calculating the number of variables in the entire model, as described for 46 sections. If we assume that each polluter can pipe its effluent to every potential treatment plant and to every river section, and each treatment plant can pipe to every river section, the number of piping variables alone would be $3(46)^2$ or 6348 variables. Since it seems reasonable from knowledge of the nature of the problem that certain of these pipes would not enter the solution, it is desirable that the nonlinear algorithm does not have to consider the flow variables associated with these pipes.

The last alteration is that our particular implementation of the model allows for tributaries. They are limited to a length of one section. This allows for polluters dumping into small tributaries some distance from the main stream. The model could easily be expanded to encompass tributaries of any number of sections in length, and even tributaries with tributaries. However, since it was not necessary for our application we limited the length of the tributaries to one section.

For the purpose of illustration of the river basin model described, we have applied the model to the West Fork White River Basin with the following restrictions:

1. All effluent must be treated at a level of at least 85 percent removal. This corresponds to required secondary treatment, which is required as a matter of policy in the West Fork White River Basin. The implication of this assumption is that no effluent can be dumped directly into the river without treatment.
2. All treatment plants must dump their effluent into the nearest river section. Since each polluter is allowed to pipe to any treatment plant, it

TABLE I—Effluent Flow

River Section	Effluent Flow c.f.s.	BOD mg/l	DOD* mg/l
4	.2700	40.0	4.66
6	20.0829	322.0	5.66
11	.300	298.0	8.66
16	24.400	200.0	6.66
23	.7800	270.0	6.66
31	13.2000	20.5	2.16
32	.9300	19.0	.36
33	13.0000	20.0	6.36
36	185.0000	450.0	6.06
37	10.0000	20.0	6.66
41	185.0000	450.0	4.76

The DOD is calculated using a saturation level of dissolved oxygen of 8.66 mg/l determined for a temperature of 21° C.

seems unnecessary to allow treatment plants to pipe to other sections.

3. An individual polluter can pipe his effluent no more than 25 river sections up or down the river. This allows for the reduction of piping variables as explained above, and since it is unlikely that piping costs will be less than gains from economies of scale over a long distance, the best possible solution will not be eliminated from consideration.
4. One potential reservoir exists at the headwaters of the river. This is not the only potential site, but was chosen for this application.
5. The quality requirement in all sections is 5 mg/l of dissolved oxygen. This is a State of Indiana policy and is, for that reason, appropriate for our problem.

The sub-model described by this set of assumptions has 1880 variables and 138 constraints. The large number of variables indicates that the appropriate use of the priority classes described in the last section of the paper is necessary. The smaller the number of variables the algorithm must examine for possible change, the faster a nonlinear iteration can take place. Therefore, the priority classes can be used to great advantage by selecting in advance the variables which appear to be the most important. This is, of course, very tricky, but the priority classes can be altered as information is obtained from the iterations of the nonlinear algorithm. In our application of the model, we selected around 250 first priority variables from our knowledge of the nature of the river problem.

The necessary data to apply the programming model to the West Fork White River is in Tables I through

TABLE II—Incremental Flow

River Section	Incremental Flow c.f.s.	BOD mg/l	DOD mg/l
1	1.0	4.0	2.36
4	-21.2	—	—
5	1.0	214.0	7.66
9	4.0	23.2	3.06
10	6.4	13.2	6.26
15	72.0	7.7	3.26
18	-35.0	—	—
19	23.0	12.1	3.06
25	13.0	5.0	2.16
26	14.0	5.0	2.76
28	-214.0	—	—
29	28.0	5.0	.76
31	5.8	12.4	2.06
35	5.0	9.3	1.26
40	8.0	13.9	2.66
41	8.0	5.0	4.76
42	9.0	5.0	2.66
43	10.0	23.2	7.66
44	21.0	16.7	7.06

IV. This is information about effluent flows, incremental flows, tributary flows and other required river parameters.

The cost functions used for the treatment plants and pipelines were obtained from the literature. The total cost function for treatment plants was obtained from Frankel¹⁰ and has the following form for the *k*th treatment plant:

$$C_k^{TP} = 49.22 \left(\sum_{i=1}^s p_{ki} \right)^{3/4} [8.0(r_k - .5)^3 + 1] \quad (30)$$

The value of p_{ki} is the flow from polluter *i* to treatment plant *k* and the value of r_k is the level of BOD removal at treatment plant *k*.

The total cost function for piping was obtained from Linaweaver and Clark¹¹ and for a particular section of

TABLE III—Tributary Flow

River Section	Effluent Flow c.f.s.	BOD mg/l	DOD* mg/l
1	51.0	2.92	.7993
2	6.1	2.63	1.9073
7	30.3	9.80	4.9465
14	3.0	5.50	3.4587
17	44.0	7.18	2.7736
24	16.10	6.91	2.2540
30	21.00	10.62	2.4544
38	61.00	30.44	4.5948
45	33.7	8.00	.3462

pipe has the form:

$$C_{ij}^P = 1.865 d_{ij}(q_{ij})^{.598} \quad (31)$$

The term d_{ij} is the length of the pipe segment and q_{ij} is the flow through that segment. If we were considering a section of pipe from polluter *j* to treatment plant *i*, the q_{ij} term would be replaced by p_{ij} . Both equations, (30) and (31), are in terms of \$1000 per

TABLE IV—River Parameters

Section	g_k	h_k	y_k	z_k	K_{1k}	X_k
1	6.579	-.249	.0445	.55	.115	5.6
2	6.579	-.249	.0445	.55	.1	.1
3	6.579	-.249	.0445	.55	.103	6.2
4	6.579	-.249	.0445	.55	.1	1.9
5	6.579	-.249	.0445	.55	.1	3.6
6	6.579	-.249	.0445	.55	.6	3.7
7	6.579	-.249	.0445	.55	.115	.1
8	6.579	-.249	.0445	.55	.63	3.2
9	6.579	-.249	.0445	.55	.63	2.8
10	6.579	-.249	.0445	.55	.63	3.0
11	6.579	-.249	.0445	.55	.623	4.3
12	.040596	.538	.0125	.728	.308	1.9
13	2.8152	-.117	.065	.471	.308	1.0
14	6.579	-.249	.0445	.55	.102	.1
15	2.8152	-.117	.065	.471	.304	2.4
16	2.9152	-.117	.065	.471	.805	12.4
17	6.579	-.249	.0445	.55	.104	.1
18	2.8152	-.117	.065	.471	.600	5.5
19	2.8152	-.117	.065	.471	.100	.5
20	.037944	.403	.0045	.715	.100	3.3
21	3.3354	-.14	.064	.448	.100	5.1
22	3.3558	-.14	.014	.685	.275	1.2
23	3.3558	-.14	.014	.685	.300	.8
24	6.579	-.249	.0445	.55	.103	.1
25	3.3558	-.14	.014	.685	.300	8.7
26	3.3558	-.14	.014	.685	.100	5.0
27	.14382	.183	.0056	.715	.100	4.8
28	.14382	.183	.0056	.715	.095	5.4
29	.003468	.645	.0023	.78	.091	5.0
30	6.579	-.249	.0445	.55	.096	.1
31	.0034	.645	.0023	.78	.093	1.8
32	.0034	.645	.0023	.78	.093	.5
33	.0034	.645	.0023	.78	.093	.3
34	.0034	.619	.0007	.935	.092	1.8
35	.0034	.619	.0007	.935	.092	.7
36	.0034	.619	.0007	.935	.201	.7
37	.0034	.619	.0007	.935	.201	.2
38	6.579	-.249	.0445	.55	.094	.1
39	.3374	.619	.0007	.935	.201	.8
40	3.621	-.197	.0050	.765	.213	5.7
41	3.621	-.197	.0050	.765	.225	3.9
42	3.621	-.197	.0050	.765	.308	7.3
43	3.621	-.197	.0050	.765	.308	8.4
44	3.621	-.197	.0050	.765	.308	2.6
45	3.621	-.197	.0050	.765	.1	.1
46	3.621	-.197	.0050	.765	.310	1.2

year and include both construction cost and operation and maintenance cost.

The reservoir costs were obtained for the particular site along with the expected flow augmentation yield. Any flow less than the expected yield is assumed to cost a percentage of the total cost equal to the percentage of total flow. The annual total cost for the construction and maintenance of a dam at the chosen site is \$807,000. The amount of expected flow in cubic feet per second is 100.

The sum of all of these cost functions for each potential structure in the river basin is the objective function of the programming model with one adjustment. The cost of every polluter treating on-site at a 85 percent removal level is subtracted from the aforementioned sum. This implies that the cost given by the objective function is the cost over and above the cost of required treatment at the 85 percent level at the polluters. We note that the required treatment level of 85 percent does not maintain the water quality at the required level of 5 mg/l.

The solution obtained to the West Fork White River problem described has the following features:

1. The effluent of the polluters in section 4 and 6 are combined and dumped into section 6.
2. The effluent to the polluters in sections 31, 32 and 33 are combined and dumped in section 31.
3. Part of the effluent of section 36, 30 cfs, is combined with the effluent of section 37 and dumped in section 39. There is a regional plane constructed at section 39 to handle the combined effluent.
4. The reservoir at the headwaters site is constructed and provides 100 cfs. of augmentation.

The rest of the polluters dump their effluent into the nearest sections after treating at the level given in Table V. The location of the polluters in the West Fork White River basin can be seen in Figure 2.

TABLE V—Treatment Levels of Treatment Plants in the Typical Solution

Section	Treatment Level
6*	.85
11	.85
16	.85
23	.85
31*	.85
36	.986
39*	.85
41	.910

*The plants in sections 6, 31 and 39 are regional plants.

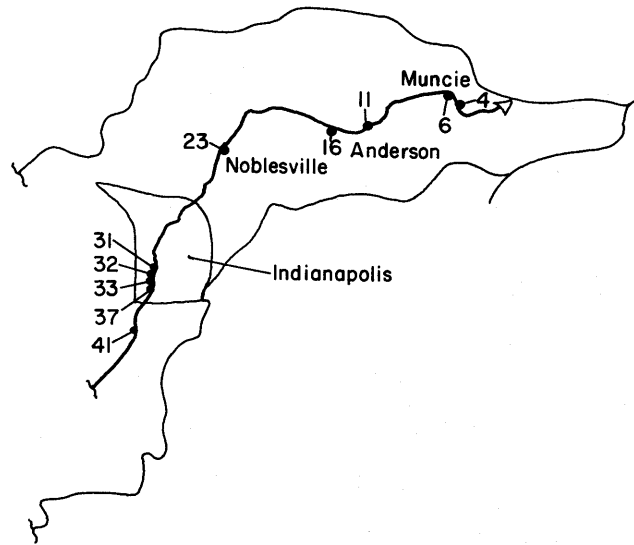


Fig. 2

The total cost of the solution obtained is \$3,571,799. This cost is over and above the cost of required 85 percent removal at all polluters. The cost of required uniform treatment of 98 percent removal over and above the cost of 85 percent removal is \$4,063,074. The required uniform treatment of 98 percent does not give a feasible solution. By combining certain effluents and using flow augmentation, the cost is reduced a half a million dollars and the river meets the water quality standards.

The solution given above to the West Fork White River pollution problem appears to be reasonable. Effluent from polluters which are close together is combined to take advantage of the economies of scale in the production of clean water. In the case of the polluter in section 36, it is necessary to transport some of the effluent downstream in order to gain feasibility. This points out a heretofore undiscussed role that piping effluent can play in solving river basin pollution problems. In the paper by Graves, Hatfield and Whinston³ on by-pass piping we see that piping replaces the treatment plant to take advantage of the natural treatment ability of the river. In our example, it is mandatory to pipe in order to meet the required quality goals, unless one can treat at a level of removal of almost 100 percent. This is practically speaking almost impossible.

ACKNOWLEDGMENTS

The development of the river simulation model in this paper owes much to the DOCAL computer program of

the Environmental Protection Agency. This program was applied on the West Fork White River by the Evansville Office of E.P.A. We are especially grateful to Max Noecker and Stanley Smith of the Evansville office who kindly made their data and experience available for our use. Discussions with our colleague J. Hamelink of the Forestry department were helpful. The authors are responsible for all possible errors.

REFERENCES

- 1 R A DEININGER
Water quality management economically optimal pollution control system
Unpublished PhD Dissertation Northwestern University Evanston Illinois 1961
- 2 D P LOUCHS C S REVELLE W R LYNN
Linear programming models for water pollution control
Management Science Vol 14 No 4 Dec 1967
- 3 G W GRAVES G B HATFIELD
A WHINSTON
Water pollution control using by pass piping
Water Resources Research Vol 5 No 1 Feb 1969
- 4 H W STREETER E B PHELPS
A study of the pollution and natural purification of the Ohio River
U S Public Health Bulletin No 146 Feb 1925
- 5 R V THOMANN
Mathematical model for dissolved oxygen
Proc Amer Soc Civil Engr 89 No SA5 Oct 1963
- 6 G W SCHAUMBURG
Water pollution control in the Delaware estuary
Harvard University Water Program Harvard University Cambridge Massachusetts May 1967
- 7 G W GRAVES G B HATFIELD
A WHINSTON
Water pollution control with regional treatment
Technical Report Federal Water Pollution Control Administration Forthcoming
- 8 G GRAVES
Development and testing of a nonlinear programming algorithm
Aerospace Corporation June 1964
- 9 G W GRAVES A B WHINSTON
The application of a nonlinear algorithm to a second order representation of the problem
Centre d' Etudes de Recherche Operationnelle Volume 11 No 2 1969
- 10 R J FRANKEL
Economic evaluation of water quality—an engineering economic model for water quality management
SERL Report No 65 3 University of California Berkeley Calif Jan 1965
- 11 F P LINAWEAVER C S CLARK
Cost of water transmission
Journal of American Water Works Association 1549 1560 December 1964

Parametric font and image definition and generation

by AMALIE J. FRANK

Bell Telephone Laboratories
Murray Hill, New Jersey

INTRODUCTION

The demand for high graphic arts quality publications particularly in the areas of education and technology increases steadily. The associated use of computer controlled photocomposition systems can be likewise expected to accelerate. Considering the high cost of currently available systems, we were motivated to investigate alternative approaches, both from a hardware and a software point of view. We conducted experiments with a high resolution electron beam recorder controlled by a small computer containing 8K words of 16 bits each, and running at about 500K cycles per second. The recorder can address a raster grid 16,384 square, and can draw both horizontal and vertical vectors, but the vectors must be no greater than four rasters apart to obtain proper shading. This paper concerns itself primarily with the software implementation, as constrained by the given hardware configuration.

Focusing on the software implementation, the means of defining and generating repeated images and fonts is a prime factor in determining the economics of a computer photocomposition system. For high quality publications, a variety of symbols and fonts with multiple sizes in each font is a necessary feature. There is, of course, a philosophic question concerning the design of new fonts oriented specifically for computer generation as opposed to the use of traditional styles, originally designed for manual stylus, woodblock, or hot metal techniques. Reserving aesthetic judgment for the time being, we concentrated on developing an efficient process for defining and generating any font or set of images, following the given lines as exactly as possible. In designing a system for this purpose, three factors must be considered: the manual operations required to arrive at the image definitions, the computer storage required to hold the definitions, and the processing time required to draw the images. For the given hardware, the computer storage condition assumes first priority. Previous software systems are found to

require excessive storage and fairly extensive manual labor to define the images. Described herein is a new method for defining and generating images in parametric form. This method decreases storage requirements and simplifies the manual operations considerably.

PREVIOUS METHODS

In most existing systems, the font definition consists of either a list of coordinates of the endpoints of the strokes comprising the images, including any internal shading strokes, or a list of the coordinates of points defining the contours of the characters. Such lists occupy a considerable amount of storage. This is further aggravated as technological advances drive the minimum line width down, thus requiring many more strokes to shade the same areas. This, of course, can be offset to some extent by including hardware to defocus the beam for larger size characters, and thus increase the beam width, but at the expense of a sharp, clear image. Another important disadvantage of existing systems is that different sizes of the same font require separate and distinct lists, themselves varying in size. For each size in a desired font, the manual operations to define the font must be repeated, and the resulting definitions stored individually.

For an example, consider a simple sans-serif font in 8 point size, and assume an average of 75 strokes per character. If the strokes all are vertical and are specified in series, then the simplest storage scheme requires for each stroke 8 bits for the starting position of the stroke relative to a local origin, and 8 bits for the length of the stroke. On this basis, a font of 128 characters requires 9,600 words of storage. This increases proportionately for larger-sized characters, and also for more complex fonts.

Some improvement to the simple encoding scheme indicated above may be made by incorporating variable length fields, or by encoding differences between suc-

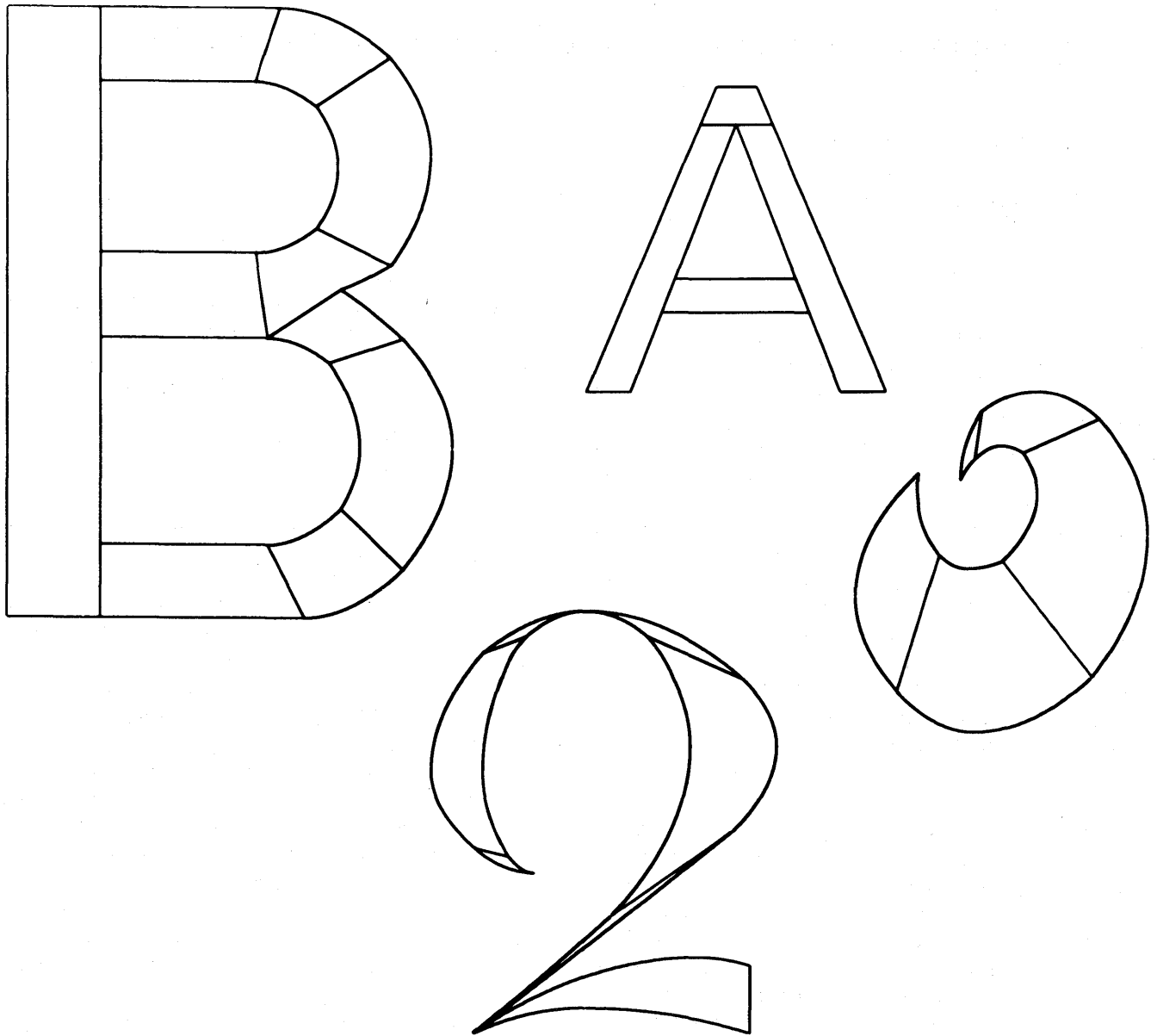


Figure 1—Patch configurations

cessive values. Other techniques arising principally from research in pattern recognition may also be considered. In one of these schemes, the boundaries of the image are "chain" encoded. Here, the position of a point is given relative to the previous point in the list, usually as a value from 0 to 7 which indicates one of the 8 raster positions immediately surrounding the previous point. Another approach introduces the concept of the "skeleton" of an image. This scheme is use-

ful in applications such as chromosome analysis where reduction of the image to a skeletal graph is an objective. However, for photocomposition, it offers no particular advantage over the other encoding schemes mentioned.

In fact none of these schemes reduces the storage requirements to a feasible level for the experimental equipment indicated previously. A new scheme which proves to be highly effective is described below.

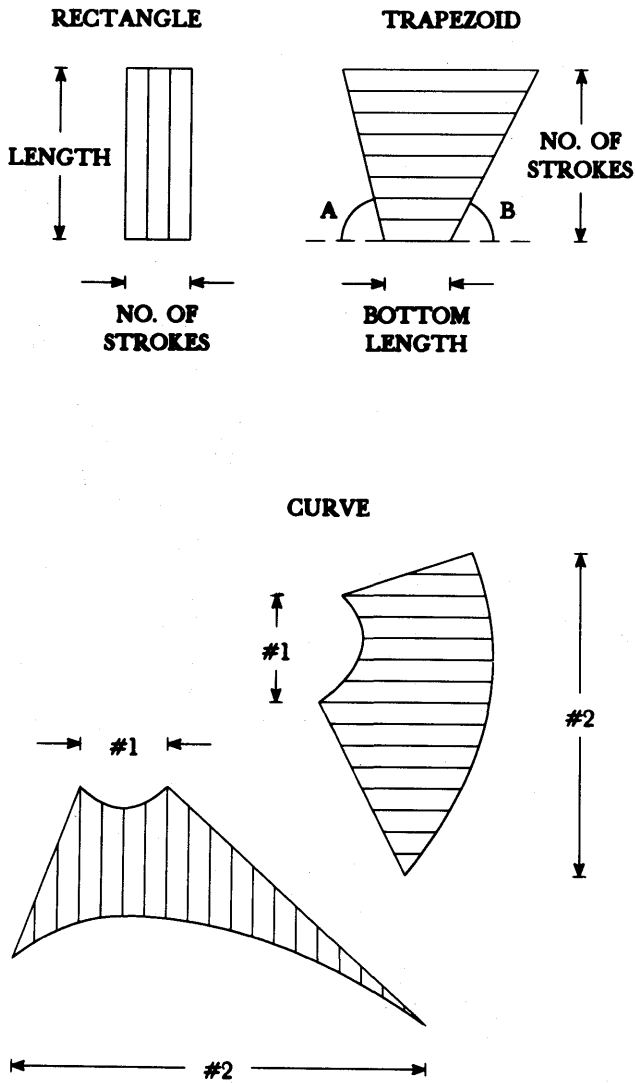


Figure 2—Types of patches

NEW IMAGE DEFINITION

This scheme uses the concept of breaking up an image into patches. The contours of each patch are then described by a set of parameters. The parameters are chosen such that a simple manipulation yields varying sizes of an image.

There are various ways in which an image can be divided up into patches, and correspondingly various ways of defining the patch parameters. One such construction was proposed by Mathews and Miller in 1965.¹ Their construction, which was designed for hardware implementation, assumed that curved portions of an image would be broken up into patches, each one of which had two curved sides and two other sides which are straight parallel lines. Each patch requires eight

parameters: the initial width, the height, the coordinates of one corner, and the curvature and slope of each curved side. Rectangles and trapezoids are treated as special cases.

The design described herein also uses the three types of patches: rectangles, trapezoids, and patches with curved boundaries. The curved patch definition is somewhat freer than that used in the Mathews and Miller construction, in the sense that it is not constrained to have two sides which are straight parallel lines. Figure 1 shows some sample patch configurations. These are described in detail further below. Of greater significance is the difference in methods used to arrive at the parameters for a particular image or set of images. Mathews and Miller used trial and error, which results in considerable effort to divide an image into patches and to determine the necessary parameters. It is particularly difficult to choose the slopes and curvatures in such a way that successive patches match well at their points of juncture, i.e., to insure against the appearance of cusps at these points. Clearly, a faster, more analytical method is necessary for producing image definitions in any quantity. This paper describes both a curved patch construction and an associated algorithm for arriving at the necessary parameters directly. The three types of patches are illustrated in Figure 2. Rectangles and trapezoids are handled in a straightforward manner. For a rectangle, the stored parameters are the width or the height, whichever is larger, and the number of strokes to be drawn. Vertical strokes are drawn if the height is greater than the width, and horizontal strokes otherwise. For a trapezoid, four parameters are stored: the bottom length, the number of strokes to be drawn, the change in abscissa of the left hand end of each successive stroke, and the change in line length for each successive stroke. These last two parameters are related, of course, to the angles *A* and *B* made by the sides of the trapezoid as shown in Figure 2.

A curved patch is a somewhat more complicated matter. We consider a curved patch to consist of two curved members. Vertical strokes are drawn where the curved members are considered as functions of *X*, and horizontal strokes are drawn where the curved members are considered as functions of *Y*. The two curved members may or may not meet each other at their end points. Where they do not meet, straight lines are assumed to connect the ends. Each curved member is described by at least one mathematical function. In some cases, a curved member is broken up into a number of segments, and each segment is described by a separate function. First, we must decide what kind of function is to be used. Second, for a given image or set of images such as the characters of a particular font,

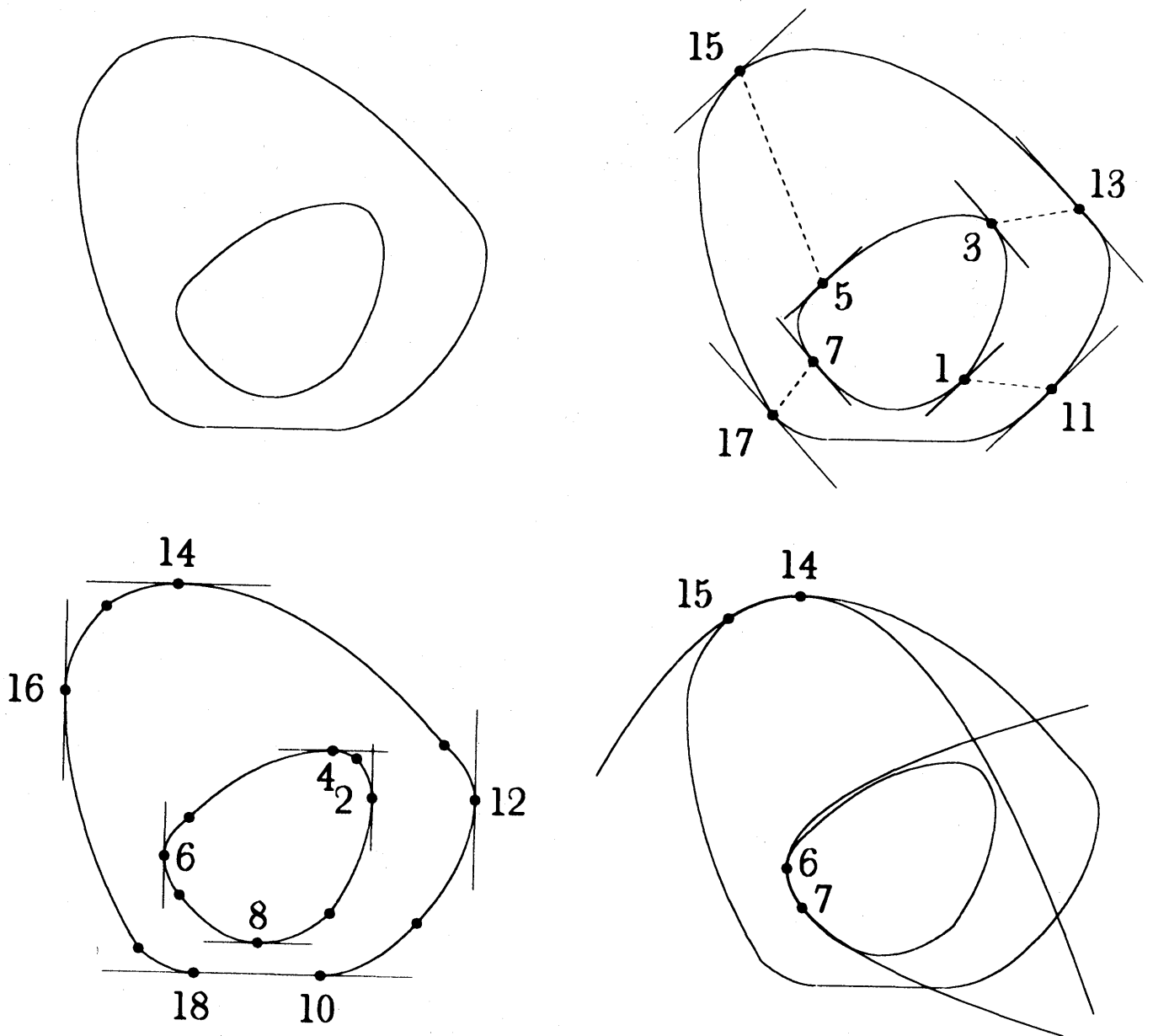


Figure 3—Steps in defining an image

we find a suitable method for determining the parameters of the function for each segment.

Two factors govern the type of function to be used: storage and execution time. A more complicated function spans a larger segment and carries through more inflections of the curve, thus requiring a smaller number of segments, and consequently less storage than a simple function. However, it takes longer to compute. Since our experiment concentrated on font production on a rather small computer, we opted to minimize execution time. For this reason, we ruled out superellipses, as suggested by Mergler and Vargo in their experiments in font design.²

We decided initially to experiment with parabolas of the type: $y = ax^2 + bx + c$ or, $x = ay^2 + by + c$. The principal axes of these parabolas are parallel to the Y and X axes respectively. Roughly speaking, we use the form $y = f(x)$ for parts of a curve that are \cup or \cap shaped, and the form $x = f(y)$ for parts that are $($ or $)$ shaped. To determine optimally which form to use where, we follow the steps illustrated in Figure 3. The upper left corner of this figure shows a donut shaped image we wish to define. We actually start the process with the donut shown in the upper right corner. Here we mark all the points where 45° and 135° lines are exactly tangent to the donut. There are eight of them.

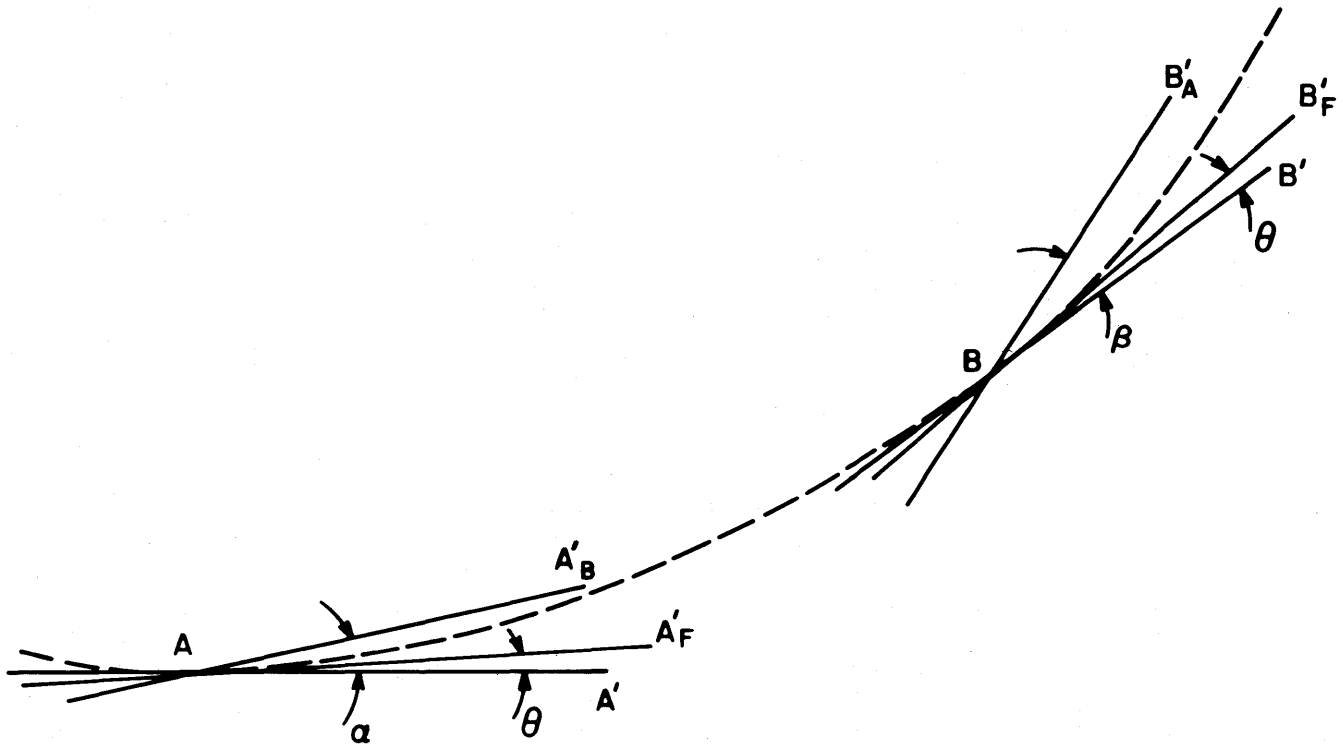


Figure 4—Equalizing the angle error

We then connect corresponding points on the inside and outside curves with dotted lines. For example, point 1 is connected to point 11, point 3 is connected to point 13, etc. This carves up the donut into four patches. For the top and bottom patches we will fit parabolas of the form $y=f(x)$, and fill the patches with vertical strokes. For the left and right patches we will fit parabolas of the form $x=f(y)$ and fill the patches with horizontal strokes.

Next we go to the lower left corner of the figure. Here we add 9 more points where 0° and 90° lines are exactly tangent to the donut. Finally, we find the parabolas which fit between successive points. We fit one parabola between points 1 and 2, another parabola between points 2 and 3, etc. The donut in the lower right corner of the figure shows two of the parabolas that have been found. Between points 14 and 15 we fit a parabola of the form $y=f(x)$, and between points 6 and 7 a parabola of the form $x=f(y)$.

GETTING A GOOD FIT

In our experiment we discovered that the process of finding the actual parameters for the parabolas to be

fit to each particular patch is not a trivial matter. We took the capital letters of the Univers font, and actually programmed a number of algorithms before we arrived at one which yields consistently good results with a minimum of manual labor. First we tried entering the coordinates of various points along the given curve. Taking the first four points, we forced the parabola to pass through the two middle points, and be a least squares fit to the two surrounding points. The resulting parabola was then used for the segment between the two middle points. We then deleted the first point, added the fifth point in sequence, and repeated the fitting process. This continued until we ran out of points. The resulting segments were fair fits, but they did not include either end segment. These could be generated by increasing the initial list by two points lying someplace on an extrapolation of the given curve member. Pinpointing that someplace proved often to be like finding a needle in a haystack.

In successive trials, we loosened up the input format by specifying for each segment separately two "fixed" points through which the parabola had to pass, and two "floating" points for the least square fit. This enabled us to place the two floating points inside or outside the two fixed points. With sufficient fishing this ap-

proach resulted in much better fits, but we were often bothered by noticeable cusps at the junction of segments. This led us to concentrate on the tangent lines of a fitted parabola at the end points. From this evolved our final algorithm, which "equalizes the angle error," i.e., makes the angle between the tangent line of the fitted parabola at one end point and the tangent line of the given curve at that point equal to the correspondingly defined angle at the other end point. This is illustrated in Figure 4 and is stated more formally as follows. Given points A and B , and their derivatives A' and B' , to fit a parabola:

1. Compute the parabola P_A which passes through A and B and has the derivative A' at A . Compute the derivative B_A' of P_A at B .
2. Compute the parabola P_B which passes through A and B and has the derivative B' at B . Compute the derivative A_B' of P_B at A .
3. The tangent lines corresponding to the two derivatives A' and A_B' at A form some angle α . Similarly, the tangent lines corresponding to the two derivatives B' and B_A' at B form some angle β . There exists a family of parabolas which pass through A and B , and which have derivatives A_F' and B_F' , whose respective tangent lines lie somewhere within the angles α and β respectively. Within this family, choose that parabola whose tangent lines at A and B "equalize the angle error," i.e., where the angle (θ) made by the tangent lines corresponding to the derivatives A' and A_F' is equal to the angle (θ) made by the tangent lines corresponding to the derivatives B' and B_F' .

This algorithm proved highly successful. We did note, however, that points on the curve where the tangent line is exactly horizontal or exactly vertical were particularly sensitive. Accordingly, all such points are made end points of curve segments. In this case alone, we do not equalize the angle error, but force the derivative of the curve at such a point to be exactly zero. In most cases this procedure gives the visual effect of continuous curvature at the juncture of adjacent segments, even though the angle made by the tangents of the two segments at the juncture differs by as much as 10 degrees. Beyond this threshold however, the contour appears disjoint. Where this occurs, the addition of one more segment normally removes this condition.

The parameters stored for a curved patch consist essentially of the coordinates of the starting and ending points of the two curved members relative to a local origin of the entire letter or image, and a set of three parameters required for each curve segment in the

patch. Various encoding economies are made, as, for example, where a patch is preceded by a contiguous patch, the end points of the previous patch are used as the starting points of the current patch. We have also incorporated a mirroring procedure so that patches which are mirror images of each other share a common set of parameters.

The curve fitting algorithm to equalize the angle error computes the coefficients a and b of the parabola as functions of the coordinates of the end points of the segment and of the tangent of the given curve at those points. The actual generation of an image, however, is not done by evaluating the function $y = ax^2 + bx + c$ or $x = ay^2 + by + c$ successively. Since the distance between strokes is constant, the computation of the functional values lends itself readily to implementation with difference equations. Classically, this approach consists of starting with an initial value of the independent variable (x_0 for vertical and y_0 for horizontal curved patches) and an initial functional value (y_0 for vertical and x_0 for horizontal curved patches). We then construct each successive functional value from the previous functional value by applying the difference equation coefficients d_0 and k , as shown below for a vertical curved patch:

$$\begin{array}{ll} y_1 = y_0 + d_0 & d_1 = d_0 + k \\ y_2 = y_1 + d_1 & d_2 = d_1 + k \\ \vdots & \\ y_N = y_{N-1} + d_{N-1} \end{array}$$

The difference equation constants are easily derived from the coefficients a and b , the initial functional value y_0 , and the inter-stroke distance h :

$$\begin{aligned} d_0 &= ah^2 + 2ahy_0 + bh \\ k &= 2ah^2 \end{aligned}$$

Use of difference equations results in less execution time since additions replace multiplications, and it results in less storage required since only two difference equation constants are required rather than the three coefficients a , b , and c . An auxiliary advantage is that the range of the difference equation constants is considerably smaller than that of the coefficients. The difference equation constants each fit within one word as fixed point numbers whereas the coefficients would probably require a floating point representation or double precision storage.

In summary, the parameters required for each curved segment are the coordinates of the starting point, the two difference equation constants, and the number of strokes spanning the segment. Figure 5 and Tables I and II contain a complete example of a parametric

TABLE I—Input Data for Image in Figure 5

POINT	X	Y	ANGLE OF TANGENT
1	799	744	90
2	744	890	135
3	594	981	165
4	440	1000	0
5	126	868	45
6	21	669	75
7	0	500	90
8	440	0	0
9	799	56	—
10	799	135	—
11	799	524	—
12	664	524	—
13	453	524	—
14	652	744	90
15	598	850	135
16	524	885	165
17	440	894	0
18	245	814	45
19	152	654	75
20	139	500	90
21	440	106	0
22	664	135	—
23	664	418	—
24	453	418	—

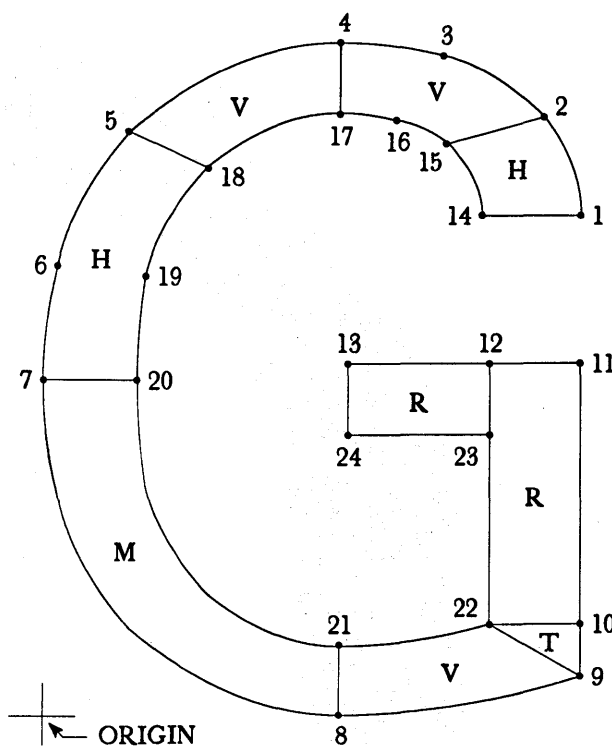


Figure 5—Example of parametric definition

definition. In Figure 5 the types of patches are coded as follows: *R*=rectangle, *T*=trapezoid, *V*=vertical curve patch, *H*=horizontal curve patch, *M*=mirror image of other patches. Figure 6 shows an exploded view of the image generated from this definition. This same definition may be used to generate the image in any other size, larger or smaller. Excessive size changes result in some degradation to the image. See Figure 7

for a few examples of sizing. In these cases, the sizing was done in a strictly proportional manner. The patches may also be individually sized by varying algorithms to obtain thinning or thickening of different parts of an image.

The various images in Figures 1, 2, 3, 5, 6, and 7 are

TABLE II—Computed Parameters for Image in Figure 5

End points of segment	Parabolic Coefficients			Difference Equation Constants	
	a	b	c	k	d ₀
1-2	-.00256	3.80424	-615.17606	-.08181	-0.04091
2-3	-.00240	2.62036	273.29839	-.07694	3.79026
3-4	-.00078	0.68705	848.84937	-.02498	0.96187
4-5	-.00136	1.19329	737.47536	-.04339	-0.02170
5-6	.00169	-2.06229	644.06991	.05411	-3.46570
6-7	.00074	-0.74405	186.01190	.02381	-0.98810
8-9	.00043	-0.38025	83.65432	.01383	0.00691
14-15	-.00446	6.63375	-1815.75331	-.14266	-0.07133
15-16	-.00414	4.21743	-188.60866	-.13243	2.92872
16-17	-.00156	1.37188	594.18601	-.04989	1.02268
17-18	-.00208	1.83257	492.83464	-.06664	-0.03332
18-19	.00218	-2.63984	943.88245	.06989	-3.66281
19-20	.00049	-0.49310	263.27415	.01578	-0.60750
21-22	.00064	-0.56122	227.46939	.02041	0.01020

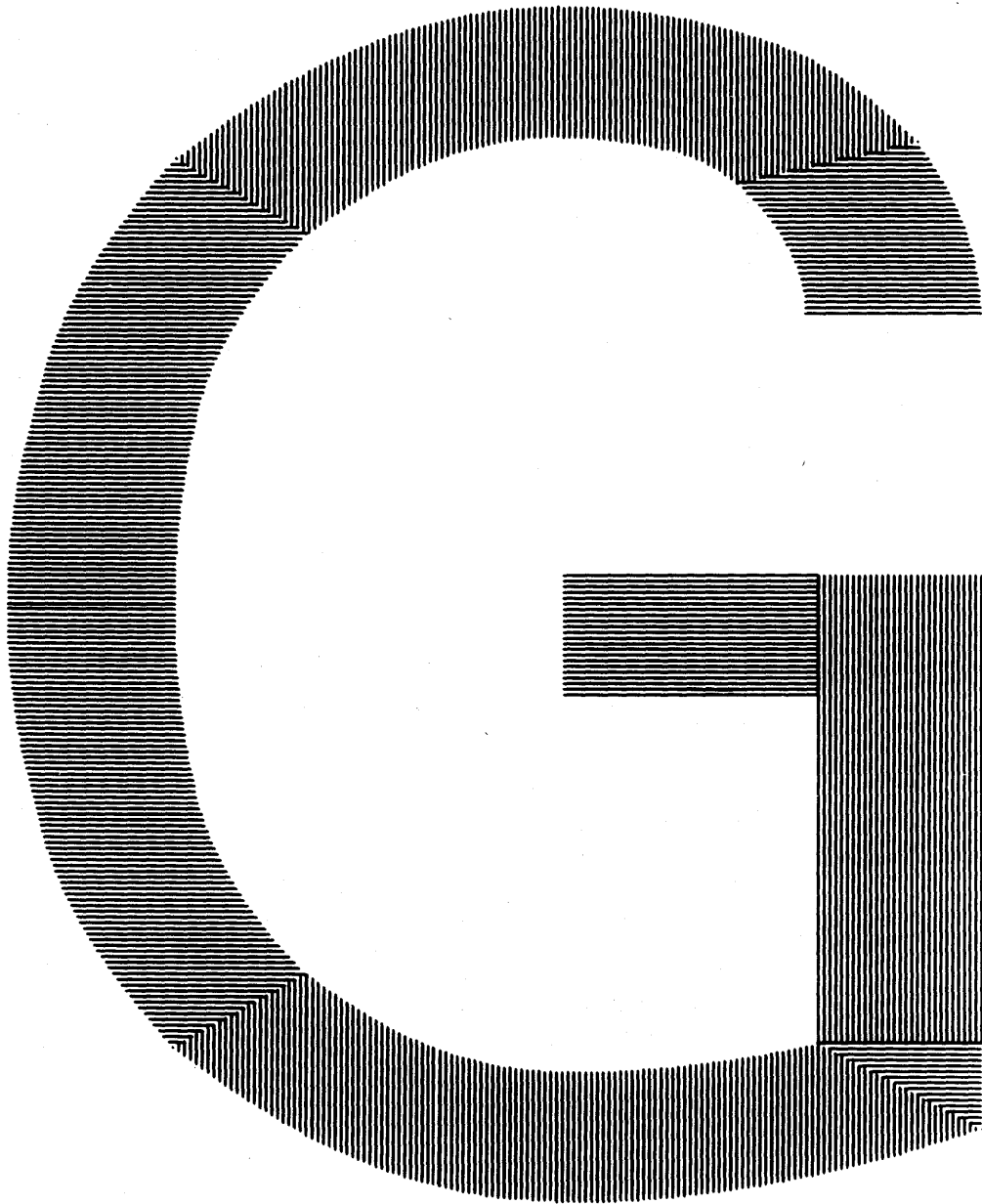


Figure 6—Exploded view of an image defined and generated by the patch method

actual computer output and were all made using this new method. The output device used for this purpose was a Stromberg-Carlson 4060 microfilm recorder.

SUMMARY AND EVALUATION

We may now evaluate the method described with respect to the manual operations, computer storage, and processing time.

The curve fitting implementation for this study was

initially done on a non-interactive basis. In this case, the manual operations consist of measuring and key-punching the coordinates of the significant points, i.e., all terminal points, inflection points, and points of tangency with 0, 45, 90, and 135 degree lines, and any additional points required for smoothing, along with the angle of the tangent line at these points. Almost all of these points are chosen to be points of tangency with 0, 45, 90, and 135 degree lines, so that it is not actually required to measure the angle in these cases. Even these manual operations are a decided improve-



Figure 7—Exploded views of various sized images derived from image shown in Figure 5

ment over previous methods, where many more points must be specified, and redone for each separate size.

This method has also been implemented in an interactive program designed by Miss Joan E. Miller of Bell Telephone Laboratories. This interactive procedure saves substantial time by replacing the measuring and keyboarding of the various coordinates and tangents by the faster physical and visual processes of drawing and manipulating various knobs, pushbuttons and switches and also by providing a means of immediate feedback and correction for the curve fitting processes. In this procedure, we draw the boundaries of the image by hand on a RAND tablet. This results in a stack of coordinates in storage, which are then subjected to a smoothing algorithm, and displayed on a scope. The next step is to find the significant points of tangency. This is done by means of a novel feature of this program, which is a tracking dot under knob control. This dot is not permitted to course the entire scope, but is constrained to follow the path of the input curve. As it does so, the tangent of the curve is continuously computed and displayed numerically on the scope. The user may thus position the tracking dot to any desired point and register it by pushing a button. After all these significant points are captured the curve fitting algorithms described previously are applied and the fitted parabolas immediately displayed. Additional segments are added as needed. Finally, the parameters describing the entire image are organized and outputted for use in the photocomposition phase.

The image definitions for this experiment were stored and executed as procedures in an interpretive language. While this adds some overhead to both the storage and processing, it affords a great degree of flexibility. For this implementation, the storage required for the 26 capital letters is approximately 700 words. Extending this to 128 characters, adjusted for smaller characters, we arrive at a conservative estimate of 3,200 words, representing a $\frac{2}{3}$ reduction over the 9,600 indicated previously. Moreover, we note that font definitions in parametric form are easily sized. This can be done in the definition process by transforming the coordinates of the defining points prior to computing the parameters, or it can be done dynamically at drawing time by maintaining a "master" set of definitions in one size and deriving other sizes as required from the master set. All sized versions of the same image require the same amount of storage, a valuable feature for efficient storage management.

Volume timing tests are currently being constructed. Initial results indicate a through-put rate in the neighborhood of 100 characters per second. This is satisfactory for many applications where graphic arts quality output is required using a general purpose computer system, and where economy of operation is a prime consideration. The speed may be increased, of course, by microprogramming or hard-wiring or by using computation equipment with a higher cycling rate.

We conclude that the method described of defining images in parametric form has a number of distinct advantages, and while it is particularly desirable in a minimum equipment configuration, it is generally applicable to a variety of hardware configurations.

REFERENCES

- 1 M V MATHEWS J E MILLER
Computer editing, typesetting and image generation
Proceedings Fall Joint Computer Conference 1965
pp 389-398
- 2 H W MERGLER P M VARGO
One approach to computer assisted letter design
The Journal of Typographic Research Volume II
Number 4 October 1968 pp 299-322
- 3 J S WHOLEY
The coding of pictorial data
IRE Transactions on Information Theory Volume IT-7
Number 2 April 1961 pp 99-104
- 4 H FREEMAN
On the encoding of arbitrary geometric configurations
IRE Transactions on Electronic Computers EC-10
Volume 2 June 1961 pp 260-268
- 5 C T ZAHN
A formal description for two-dimensional patterns
Proceedings of the International Joint Conference on
Artificial Intelligence May 7-9 1969 Washington DC
Gordon and Breach NY 1969
- 6 U MONTANARI
*A note on minimal length polygonal approximation to a
digitized contour*
Communications of the ACM Volume 13 Number 1
January 1970 pp 41-47
- 7 J L PFALTZ A ROSENFELD
Computer representation of planar regions by their skeletons
Communications of the ACM Volume 10 Number 2
February 1967 pp 119-125
- 8 U MONTANARI
Continuous skeletons from digitized images
Journal of the ACM Volume 16 Number 4 October 1969
pp 534-549

A syntax-directed approach to pattern recognition and description

by LARRY D. MENNINGA

Western Washington State College
Bellingham, Washington

INTRODUCTION

Pattern recognition has held the attention of researchers for quite some time. Early efforts were in optical character recognition and were intended to provide easier and more rapid communication from man to machine. In more recent years, this research has expanded to include the processing of pictorial information, such as that from high energy physics or medical research.

Until recently, most of the work and associated theory has treated pattern recognition as a classification or categorization problem. The methods used can be broadly characterized as follows: Each sample pattern is represented by an n -dimensional vector whose components are the values of the individual features, or properties, which have been selected for measurement. The classification is done by partitioning n -space, referred to as the feature space, into subspaces. Each subspace represents a class, and a sample pattern is considered to be a member of the class corresponding to the subspace of which the pattern vector is a member. The members of each class must be clustered in the feature space to achieve successful recognition. The use of weight vectors, proper selection of features, adaptive systems, and other techniques for improvement have been investigated. A survey of this work is given by Nagy.¹

The most significant shortcoming of the above method is that the result of such a recognition process yields only a classification, a class name or number. What is often desired is a structural description of the pattern or an analysis of the relationships existing between certain substructures within the pattern. This is especially true of more complex patterns. Of course, it is possible to divide the feature space into classes, with each one corresponding to a different structural description. However, complex patterns would necessitate an extremely large number of classes and would result in unmanageable problems.

Such classification methods are also essentially one-level. The features selected for measurement must be able to detect any structural relationships which are significant. As the patterns become more complex, the selection of a suitable set of features becomes more difficult.

Finally, the vector representation of a pattern is not the most natural for people. To make use of the potential for powerful recognition systems using interactive computing, the human being must be able to communicate with the computer using representations which he can grasp rapidly.

The linguistic approach

In recent years, research has been done using linguistic methods to overcome the failures of the classification methods. The linguistic methods are aimed specifically at producing structural descriptions of patterns. Miller and Shaw² give a survey of much of this work.

In the linguistic approach, the pattern, or picture, is considered to be a sentence in a language generated by a given grammar. This grammar is defined either explicitly or implicitly, and it is used to analyze each sample pattern. In such a grammar, the nonterminal symbols are phrases descriptive of a subpattern and the terminal symbols (called primitives) are the basic elements which are given *a priori*, and they are recognized by some method outside of the linguistic model.

A given sample pattern is then described, or recognized, by using the grammar to analyze it. A derivation tree gives a description of the sample in terms of substructures and the relationships which they satisfy—insofar as these are included in the grammar. Thus, in addition to being the desired result of an analysis, the description, as expressed in the rules for the grammar, can also direct the recognition process by defining the sequence of algorithms to be used.

```

<pattern rule>::=<object> → <description>
<description>::=<compound object> <predicate list>
<compound object>::=<basic description>|<basic descrip-
tion> <compound object>
<basic description>::=<basic object>|<basic object>
<relator> < basic object>
<basic object>::=<object>|(<description>)
<object>::=<object type>(<object label>:<constituent
list>)
<object type>::=<identifier>
<object label>::=<identifier>
<constituent list>::=<object label>|<object label>,
<constituent list>
<relator>::=<predicate name>|<predicate name> <relator>
<predicate list>::=NULL|<predicate>|<predicate> <predicate
list>
<predicate>::=<predicate name>[<constituent list>]
<predicate name>::=below|above|leftof|rightof|parallel|
perp|skew|equal|tangent|intersect|
near|far|equallength

```

Figure 1—Syntax for the pattern rules

A SYNTAX-DIRECTED SYSTEM

A system called PARSE, which uses the linguistic approach to pattern description, will be described here. PARSE is an acronym for **P**attern **A**nalysis and **R**ecognition by **S**yntax **E**valuation. It is a system in which the user must supply the metalanguage to be used to analyze and describe patterns.

Structure of grammar for PARSE

The formal syntax for a user-supplied grammar rule is shown in Figure 1. Each rule in the grammar is called a *pattern rule* and gives the definition of an object. A set of pattern rules is a *pattern grammar*. The left side of each rule begins with an identifier. This identifier is called an *object type* and is a nonterminal symbol in the vocabulary of the pattern grammar. In addition, the left side of each rule includes a list of labels.

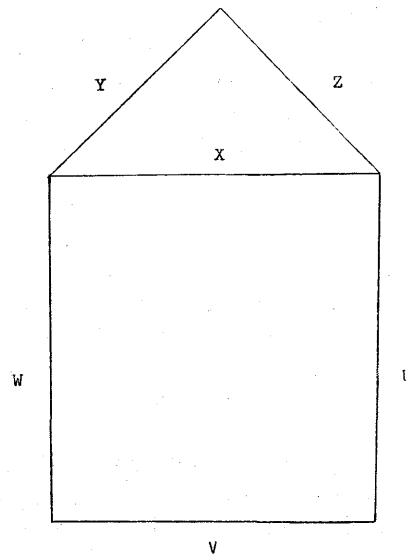
The right side of each rule consists of a list of object types, with associated labels, and predicates. There are two special object types, POINT and LINE, which are terminal symbols in the vocabulary. These are the primitives, that is, they are object types which have been defined *a priori*. They are the basic elements in the language (or in the patterns), and they are recognized outside of the PARSE system. The list of predicates in Figure 1 is not intended to be exhaustive, but rather to represent a typical set.

More than one pattern rule is allowed for any object type. This makes it possible to give alternate definitions for an object type. By using more than one pattern rule, it is possible to construct recursive definitions. Each

object type must be defined by a pattern rule, or be a primitive, if it is to appear on the right side of a pattern rule.

As an example, consider the grammar rules given in Figure 2. In the example, the object type HOUSE is defined in terms of the object types TRIANGLE and RECTANGLE and the relationships are specified by the predicates *above*, *parallel*, *perp*, and *skew*. The labels are used to identify object types, and they may be used to specify a correspondence between substructures. The relationship of identity is given implicitly by repeating an object label more than once in a pattern rule. In the rule for HOUSE, the object label X is associated with both TRIANGLE and RECTANGLE to indicate that the same instance of the line X must be a part of each of these objects.

Although the user may specify the grammar which he wants to be used, this specification is subject to certain restrictions imposed by the syntax rules of Figure 1. In each pattern rule, the labels and predicates specify semantic information for the rule, and so these elements can be treated separately. Ignoring this semantic information, the “underlying” grammar can



```

HOUSE(H:T,S) → TRIANGLE(T:X,Y,Z) above RECTANGLE(S:X,U,V,W)
TRIANGLE (T:X,Y,Z) → LINE(X:P,Q) LINE(Y:Q,R) LINE(Z:R,P)
skew[X,Y]
RECTANGLE(R:X,Y,Z,W) → LINE(X:P,Q) parallel LINE(Z:U,V)
LINE(Y:Q,U) parallel LINE(W:V,P)
perp[X,Y]

```

Figure 2—Sample pattern grammar

easily be seen to be a context-free phrase structure grammar, as defined by Chomsky,³ since only one nonterminal symbol is allowed to appear on the left side of a pattern rule. Note that the empty string cannot be generated by any of the allowable rules and that all rules are length preserving. For such a context-free grammar, there is a derivation for every sentence in the language which is generated by that grammar.⁴ Thus it is possible to define an algorithm to determine a description (structure) for a given pattern (string) in the language.

Semantics used by PARSE

Knowing the structure of a pattern is not enough. Semantics are also involved in the descriptions of patterns. The semantics give meaning (principally spatial relationships) to a particular structure. Semantic information is given primarily by the predicates and the object labels in the pattern rules. The meaning supplied by these elements of the language will be formalized in a manner similar to that suggested by Knuth.⁵

The semantics will be supplied by the values of certain attributes, or properties, that the symbols of the language will have. The attributes selected can be divided into two areas: geometric properties and labels. The value of each attribute is assigned by evaluating a function. The same name will be used for an attribute and its corresponding function.

Each pattern rule has semantics associated with it. Thus it is necessary to have a function for each attribute of each symbol in the rule. The value assigned to an attribute of a given symbol depends on the values of some of the attributes of the other symbols in the rule and on some of the other attributes of the same symbol.

Consider the underlying grammar given by a set of pattern rules. Rules in the underlying grammar will be called *syntax rules*. Let G be such a context-free grammar, $G = (V_T, V_N, P, S)$, where $V_T = \{\text{terminal symbols}\}$, $V_N = \{\text{nonterminal symbols}\}$, $P = \{\text{syntax rules}\}$, and $S = \text{the start symbol}$. For each symbol X in $V_T \cup V_N$, there is a finite set of attributes or properties, $A(X)$. Let Y_α be the set of values that can be assumed by a given attribute α in $A(X)$.

If the r th syntax rule is

$$X_0 \rightarrow X_1 X_2 \dots X_n$$

where $X_0 \in V_N$ and $X_j \in V_T \cup V_N$ for $1 \leq j \leq n$ then the semantics can be defined as follows: For each attribute α of a symbol X_j , there is a function $f_{j\alpha}$ which maps the values of certain attributes of X_0, X_1, \dots, X_n into a value of α . That is, $f_{j\alpha} : Y_{\alpha_1} \times Y_{\alpha_2} \times \dots \times Y_{\alpha_t(j,\alpha)} \rightarrow Y_\alpha$, where $\alpha_i = \alpha_i(j, \alpha)$ is an attribute of X_{k_i} , for $0 \leq k_i = K_i(j, \alpha) \leq n$ and $1 \leq i \leq t(j, \alpha)$.

$$A_g(X) = \{\text{angle, length, } x \text{ min, } x \text{ max, } y \text{ min, } y \text{ max}\}$$

for every symbol X which is not a primitive.

$$A_g(\text{POINT}) = \{x \text{ min, } y \text{ min}\}$$

$$A_g(\text{LINE}) = \{\text{angle, length, } x \text{ min, } x \text{ max, } y \text{ min, } y \text{ max}\}.$$

Figure 3—Attributes of the object types

For each symbol X , partition $A(X)$ into two disjoint subsets $A_g(X)$, the geometric attributes, and $A_L(X)$, the attributes dealing with the labels. The sets of geometric attributes are given in Figure 3. The label attributes are handled in a similar fashion.

Each attribute is assigned a value by a function of the other attributes. Let $((x_1, y_1), (x_2, y_2))$ be a pair of points giving the cartesian coordinates of the primitive LINE and (x, y) be the coordinates for POINT. Then the functions which give the semantic rules are defined as follows:

$$\text{angle}(\text{LINE}) = \begin{cases} \arctan[(Y_1 - Y_2)/(x_1 - x_2)] & \text{for } x_1 \neq x_2 \\ \pi/2 & \text{for } x_1 = x_2 \end{cases}$$

$$\text{length}(\text{LINE}) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

$$x \text{ min}(\text{POINT}) = x$$

$$y \text{ min}(\text{POINT}) = y$$

$$x \text{ min}(\text{LINE}) = \text{minimum } \{x_1, x_2\}$$

$$x \text{ max}(\text{LINE}) = \text{maximum } \{x_1, x_2\}$$

$$y \text{ min}(\text{LINE}) = \text{minimum } \{y_1, y_2\}$$

$$y \text{ max}(\text{LINE}) = \text{maximum } \{y_1, y_2\}$$

For the r th rule $X_0 \rightarrow X_1 X_2 \dots X_n$ the semantic rules for the attributes in A_g are:

$$\text{angle}(X_0) = \sum \text{angle}(X_i) \cdot \text{length}(X_i) / \sum \text{length}(X_i)$$

$$\text{length}(X_0) = \sum \text{length}(X_i)$$

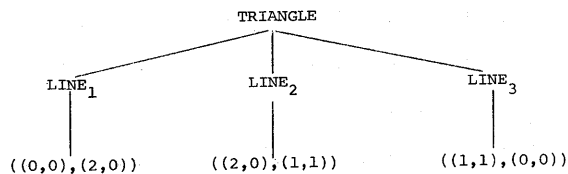
$$x \text{ min}(X_0) = \text{minimum } \{x \text{ min}(X_i) \mid i = 1, 2, \dots, n\}$$

$$x \text{ max}(X_0) = \text{maximum } \{x \text{ max}(X_i) \mid i = 1, 2, \dots, n\}$$

$$y \text{ min}(X_0) = \text{minimum } \{y \text{ min}(X_i) \mid i = 1, 2, \dots, n\}$$

$$y \text{ max}(X_0) = \text{maximum } \{y \text{ max}(X_i) \mid i = 1, 2, \dots, n\}$$

A "meaning" is assigned to each sentence in the language by the semantics. A derivation of the sentence is carried out in the usual way, using the syntax rules. Starting with the terminal symbols, the attributes are evaluated for each symbol in the derivation. It is easy to see that the semantic rules define the attributes of a symbol X as a function of the attributes of those symbols which appear on the right side of a production



object	angle	length	x min	x max	y min	y max
LINE ₁	0	2	0	2	0	0
LINE ₂	$\pi/4$	2	1	2	0	1
LINE ₃	$\pi/4$	2	0	1	0	1
TRIANGLE	0	2+2+2	0	2	0	1

Figure 4—Semantic evaluation of TRIANGLE

defining X . Thus, by first evaluating the attributes of the terminal symbols, and working backward through the derivation, the attributes of each symbol can be defined. When all the attributes can be evaluated the semantic rules are said to be *well-defined*. The meaning of the sentence is the value of the attributes of the start symbol.

In addition, any symbol can be considered to have meaning, determined by the values of its attributes. The PARSE system allows restrictions to be placed upon the meaning of certain symbols or sub-derivations. These restrictions are used to limit membership in a given pattern language. Thus, if P_G is a pattern grammar with underlying grammar G , then $L(P_G)$, the language generated by P_G , is just those sentences with structure generated by G which satisfy the semantic restrictions imposed by the predicates and labels.

The specifications of the syntax for the pattern rules allow predicates to be used in two ways. When a predicate is used as an instance of <relator> it is a two argument predicate, with the first argument being the basic object preceding it, and the second argument the basic object that follows the predicate. When a predicate is used as an instance of the metalinguistic variable <predicate list>, its arguments are the objects corresponding to the labels appearing as formal parameters in the rule. In both cases the predicate is evaluated using the geometric attributes of the object types which make up its arguments. The predicates are defined to allow sets of symbols as arguments because instances of <basic object> can include more than one symbol.

To illustrate how the semantic information is

evaluated and used, consider the grammar rules given in Figure 2. For each symbol in the grammar, the set of geometric attributes is the same. Thus $A_g(\text{HOUSE}) = A_g(\text{TRIANGLE}) = A_g(\text{RECTANGLE}) = \{\text{angle, length, } x \text{ min, } x \text{ max, } y \text{ min, } y \text{ max}\}$. Figure 4 shows the derivation of a sample TRIANGLE and a table of attribute values using the given pattern rule. The underlying syntax rule for TRIANGLE is:

$$\text{TRIANGLE} \rightarrow \text{LINE}_1 \text{ LINE}_2 \text{ LINE}_3.$$

Using the grammar

Recognition

The analysis uses the techniques of syntax-directed compiling⁶ to recognize patterns. A "top down" analysis of each sample pattern is done to determine if it is a sentence in the language generated by the grammar. A sketch of the procedure to use is as follows: Consider the input pattern to be a set, Q , rather than a string. This set is assumed to be finite. Now, beginning with the start symbol, S , or global object type, generate a sentence, or pattern, by first replacing the start symbol by its definition as supplied by a production with that symbol as the left side: $S \rightarrow X_1 X_2 \dots X_n$.

- (1) If X_1, X_2, \dots, X_n are all terminal symbols, then map the set $\{X_1, X_2, \dots, X_n\}$ into the input set, Q . Evaluate the semantics for the production using the values of the attributes of the images of the X_i 's. If the semantic restrictions are met, then Q is an instance of the object type S , and it is a member of the language generated by S . If the semantic restrictions are not satisfied a new mapping must be tried. This continues until either all the possible mappings are tried or the semantics are satisfied.
- (2) If the X_i 's are not all terminal symbols, then choose the first nonterminal symbol, say X_j . Apply the first production for X_j (if there is more than one production): $X_j \rightarrow X_{j1} X_{j2} \dots X_{jm}$. Now proceed as in step (1) above, mapping $\{X_{j1}, X_{j2}, \dots, X_{jm}\}$ into Q .
- (3) Continue, by repeating steps (1) and (2) until all terminals are reached. If at any point in the procedure all the mappings have been tried and the semantic restrictions have not been met, then go back to the previous performance of step (2) and apply the next alternate definition. If all the alternate definitions for a given nonterminal symbol have been tried for a particular performance of step (2), then go to the step (2)

two levels back. If all the alternate definitions for the start symbol have been tried and the semantics are still not satisfied, then the input set, Q , is not in the language generated by S .

Description

Since the primary motivation for the linguistic approach to pattern recognition is to produce a description of the patterns processed, the result of an analysis must allow this possibility. The result of a parse, as presented in the above, will be a yes or no answer, as to whether the input pattern is an instance of the object type which is the start symbol of the pattern grammar being used.

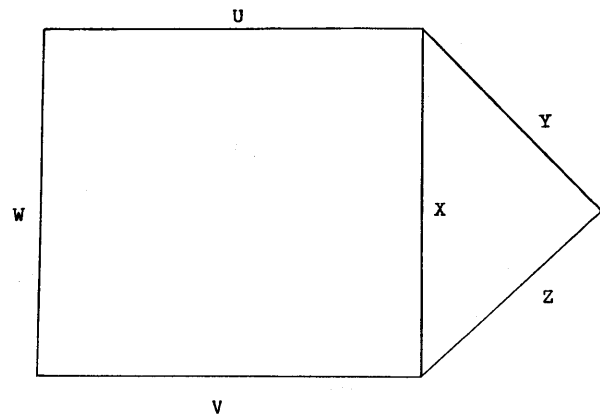
In addition, the derivation of the sentence should be kept, so that the user can have this derivation printed out. This will then be a description of the pattern in terms of the subobjects of which it is composed and the spatial relationships which they satisfy, as specified in the pattern rules.

This is all very good and useful, but it does not give any information about patterns which are not sentences in the pattern language. A description of such a pattern, in terms of object types within the grammar and the spatial predicates, should also be produced.

The object types, used as nonterminal symbols in the pattern grammar, form a natural hierarchy. The start symbol is the highest level object type in that hierarchy. The object types are ordered by assigning X a lower level than Y if the first production defining X occurs previous to the first production defining Y . Because the object types in the right side of a pattern rule must all be defined, this will result in having X lower level than Y if there is a production $Y \rightarrow sXt$ and no production $X \rightarrow uYv$, where $s, t, u,$ and v are strings of symbols (possibly empty). Also, X is lower level than Y if Z is lower level than Y and $Z \rightarrow sXt$ occurs before any production of the form $X \rightarrow uZv$.

In cases in which the input pattern is not a sentence in the language, the desired result is a description of the input in terms of the highest level object types. This description will not be unique, in general, but should be minimal in some sense.

Minimization will be achieved as follows: First the highest level object will be found such that the input pattern is a candidate for inclusion in the language generated starting with that object type. In other words, the highest level object type, such that the input includes an instance of it, is found first. Each input line, used to compose the object found, is marked. Now an instance of the highest level object type in the hierarchy, which is composed of the maximum number of un-



TRIANGLE(T:X,Y,Z) rightof RECTANGLE(S:U,X,V,W)

Figure 5—Maximal description of a pattern

marked lines, is sought. If none is found with at least one unmarked line, the object type which is the second highest in the hierarchy must be tried. If an instance is found, the unmarked lines are marked, and another instance of that type is sought. This process continues until all lines are marked or all object types have been tried unsuccessfully.

A list of the instances of object types found in this manner is constructed. The predicates defined within the PARSE system are evaluated using all pairwise combinations of objects found as arguments. Those predicates which are true are added to the list of object types, and this then becomes the description of the input pattern. Figure 5 is an example of such a description.

COMPUTER IMPLEMENTATION

Computer and language used

PARSE has been programmed on the Burroughs B5500 computer system at the University of Washington Computer Center. Interaction is provided by using a remote teletype as an input and output device. Consideration was given to using a list processing language for the implementation, since much of the data is best handled by using linked-list data structures. However, for reasons of availability, as well as ease of programming, Burroughs Extended ALGOL was used.

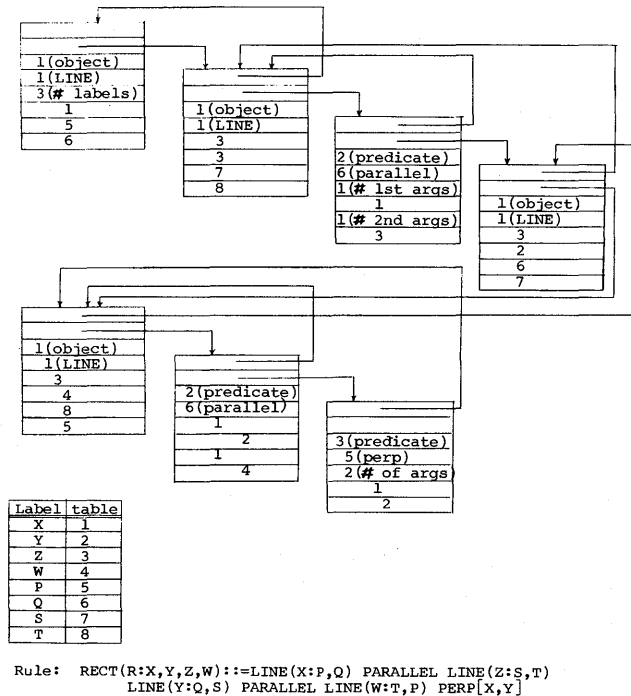


Figure 6—Internal representation of a pattern rule

Program structure

The PARSE program can be divided into three major functional areas:

1. Preprocessor for pattern grammar rules.
2. Building of the data structure.
3. Analysis of the data according to the pattern grammar.

The pattern rules are input from the teletype according to the syntax of Figure 1. The pattern rules are checked for syntactical correctness by a top-down analysis. Each pattern rule is made into a doubly-linked list structure. An example of such a structure is shown in Figure 6.

Each node in the list contains two pointers. The forward pointer indicates the next node in the definition, while the backward pointer specifies the node to go back to in case backtracking is necessary during the analysis of a pattern. Each node has a flag to indicate whether it is a predicate or an object type, the name of the predicate or object type, and a list of integers specifying the labels used by that object type or predicate.

The data structure

The pattern is also represented by using a linked-list structure. This representation arises naturally from the

hierarchical treatment of the data during analysis. Each node represents an instance of an object type and consists of several fields. These are a label, a type designation, a pointer to the next node of the same type object, and pointers to several lists. There is a subobject list made up of objects which compose the given object and also a superobject list which contains those objects of which it is a part. There is also an attribute list which contains the values of the geometric attributes the object has.

Before any processing of the pattern is done, the data structure consists of only lines and points, the primitives. During the analysis, if a given object type is being sought, the data structure is examined to see if an instance of that type object is present. If not, the rule defining that object is invoked. Each node of the definition is to be satisfied by searching the data or invoking a definition if the node specifies an object type, or by evaluating the predicate if the node specifies one.

As each object type is found it is inserted into the data structure. Thus, once a given instance of an object has been found, by satisfying the pattern rule defining it, the work done in finding it will not need to be repeated even though it may not be used at that point in the analysis.

CONCLUSION

PARSE is most similar to the system described by Evans.⁷ Evans' grammars are somewhat more restrictive in their specification of semantics. The Picture Description Language of Shaw^{8,9,10} can be modeled in PARSE by associating a label for head and a label for tail with each object type. Similarly, the system of Ledley^{11,12} is less powerful than PARSE. The only spatial relationship that his system allows is concatenation, which can be handled with the labels alone in PARSE.

While some simple patterns have been processed using PARSE,¹³ it has not been tested on any complex pictures, and thus performance figures are not available. In order to handle any automatic picture processing, a device for recognizing the primitives would be a necessary addition to the system.

The PARSE system produces a description of an input pattern in terms of a meta-language supplied by the user. Although "natural" is a subjective judgment, the description must be termed natural, in that it is symbolic, using a familiar vocabulary with its usual meaning. The PARSE system allows interaction between man and machine. Success at recognizing any instance of a pattern defined by the grammar is guaranteed by the restriction that the grammar be context-free. Further, a description of any picture will always

be produced, although not a unique one, or necessarily the same description that a human being would give.

REFERENCES

- 1 G NAGY
State-of-the-art in pattern recognition
Proceedings of the IEEE 56-5 836-62 May 1968
- 2 W F MILLER A C SHAW
Linguistic methods in picture processing—A survey
Proceedings AFIPS 1968 Fall Joint Computer Conference
Thompson Book Co Washington 1968
- 3 N CHOMSKY
Formal properties of grammars
In Handbook of Mathematical Psychology Vol II
Luce Bush Galanter Eds Wiley New York 1963
- 4 J E HOPCROFT J D ULLMAN
Formal languages and their relation to automata
Addison Wesley Menlo Park California 1969
- 5 D E KNUTH
Semantics of context free languages
Mathematical Systems Theory 2-2 127-145 June 1968
- 6 T E CHEATHAM JR K SATTLEY
Syntax-directed compiling
Proceedings of the AFIPS Spring Joint Computer
Conference Spartan Books Inc Washington 1964
- 7 T G EVANS
A grammar controlled pattern analyzer
Proceedings of the IFIP Congress Edinburgh 1968
- 8 A C SHAW
The formal description and parsing of pictures
Report No 84 Stanford Linear Accelerator Center
Stanford California 1968
- 9 A C SHAW
*A formal picture description scheme as a basis for picture
processing systems*
Information and Control 14 9-52 January 1969
- 10 A C SHAW
Parsing of graph representable pictures
JACM 17-3 453-81 July 1970
- 11 R S LEDLEY ET AL
*FIDAC Film input to digital automatic computer and
associated syntax directed pattern programming system*
In Optical and Electro Optical Information Processing
J Tippep et al Eds MIT Press Cambridge Mass 1965
- 12 R S LEDLEY
High speed automatic analysis of biomedical pictures
Science 146 October 1964
- 13 L D MENNINGA
*A syntax-directed approach to the recognition and
description of visual images*
Tech Rep TR 70-10-06 University of Washington
Seattle 1970

Computer pattern recognition of printed music*

by DAVID S. PRERAU

*Department of Transportation/Transportation Systems Center
Cambridge, Massachusetts*

INTRODUCTION

A major area of concentration of pattern recognition research has been the design of computer programs to recognize two-dimensional visual patterns. These patterns may be divided into two classes:¹ patterns representing objects in the real world (e.g., landscapes, blood cells) and patterns representing conventionalized symbols (e.g., printed text, maps). The standard notation used to specify most instrumental and vocal music forms a conventionalized, two-dimensional, visual pattern class.

This paper will discuss computer recognition of the music information specified by a sample of this standard notation. An engraving process is generally used to produce printed music, so the problem can be termed one of computer pattern recognition of standard engraved music (though the recognition procedure will, of course, be effective for music printed by any method).

The overall process is illustrated in Figure 1. A sample of printed music notation is scanned optically, and a digitized version of the music sample is fed into the computer. The digitized sample may be considered the data-set sensed by the computer. The computer performs the recognition and then produces an output in the Ford-Columbia music representation. Ford-Columbia is an alphanumeric language isomorphic to standard music notation. It is therefore capable of representing the music information specified by the original sample. (An example is shown in Figure 2.) In the form of a Ford-Columbia alphanumeric string, the output of the program can be used as input to music analysis programs,² music-playing programs, composer aids, Braille-music printers, music displays, commercial music printers, etc.

* This paper is based on a thesis submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at the Massachusetts Institute of Technology, Department of Electrical Engineering, September 1970; the work was supported in part by the National Institutes of Health.

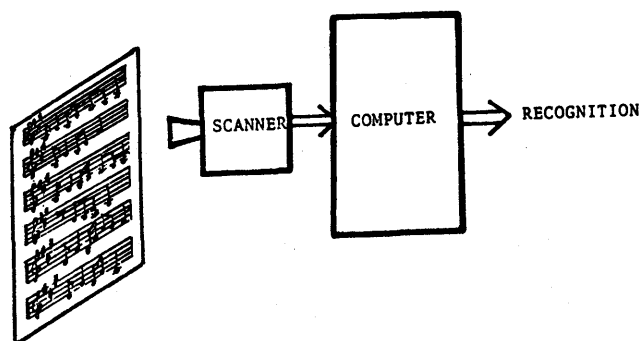


Figure 1—The overall process

THE PROGRAM

The product of this research is a computer program which recognizes standard engraved music notation. The program is called the Digital Optical-Recognizer of Engraved-Music Information, or DO-RE-MI.

DO-RE-MI is written using two computer languages, MAD and SLIP. MAD is a FORTRAN-like language, and SLIP is a list-processing language. The version of SLIP used in this study is embedded in MAD, and the combination of the two languages may be considered an extended MAD language with list-processing capability. DO-RE-MI was programmed on the MIT Compatible Time-Sharing System (CTSS), utilizing the IBM 7094 computer.

The set of symbols used in standard music notation is large. Therefore, it was decided that DO-RE-MI would be designed to recognize only a subset of these symbols; but a subset that would include all the more important symbols of music notation. This allows the program output to have practical utility, since many applications do not require full recognition. (For example, an analysis of the statistical frequency of occurrence of different pitch intervals would require only recognition of notes, clefs, accidentals, and key-signatures.) Moreover, the program is designed to be

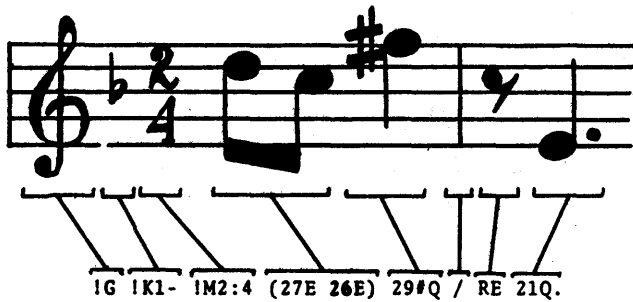


Figure 2—An example of the Ford-Columbia music representation

modular. Thus, it should not prove too difficult to add to DO-RE-MI, if desired, the capability of recognizing any of the secondary notational parameters not now recognized. This is in contrast to a previous work on computer recognition of printed music by Pruslin³ which dealt with only a small subset of music notation and was not readily extendable to the remaining notational symbols.

DO-RE-MI is divided into three sections: Input, Isolation, and Recognition. In brief, the Input Section inputs a sample of standard engraved music notation to the computer, the Isolation Section isolates the

notational symbols, and the Recognition Section performs the recognition. A flow-chart of the program is shown in Figure 3.

INPUT

The Input Section takes a sample of music notation and, using a flying-spot scanner, digitizes the sample. Several samples of source music were selected, and a positive transparency made for each. Each sample was chosen to contain two to three measures of duet music. Such a sample will be called a "picture". The scanner used is SCAD,⁴ which was developed by Dr. Oleh Tretiak at M.I.T. SCAD will scan a transparency, measure the light transmission at a large number of points on the transparency, and convert these measurements to digital form.

For each picture, SCAD scans a raster of 512 rows by 512 columns, an inch in the original sample corresponding to about 225 points. SCAD finds a 3-bit number, *T*, corresponding to the transmittance of light through each raster point. The original picture contained, at least ideally, only black regions and white regions. It is therefore reasonable to compress the data

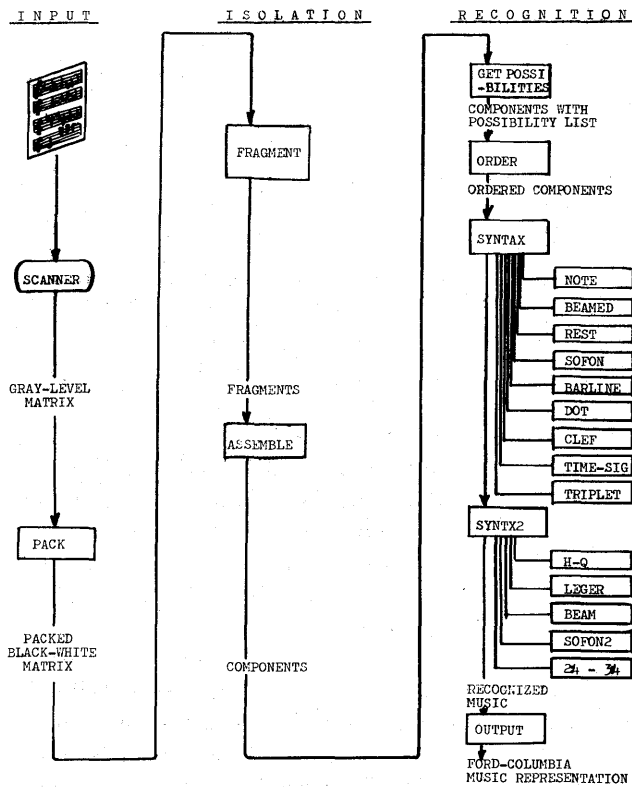


Figure 3—DO-RE-MI flowchart

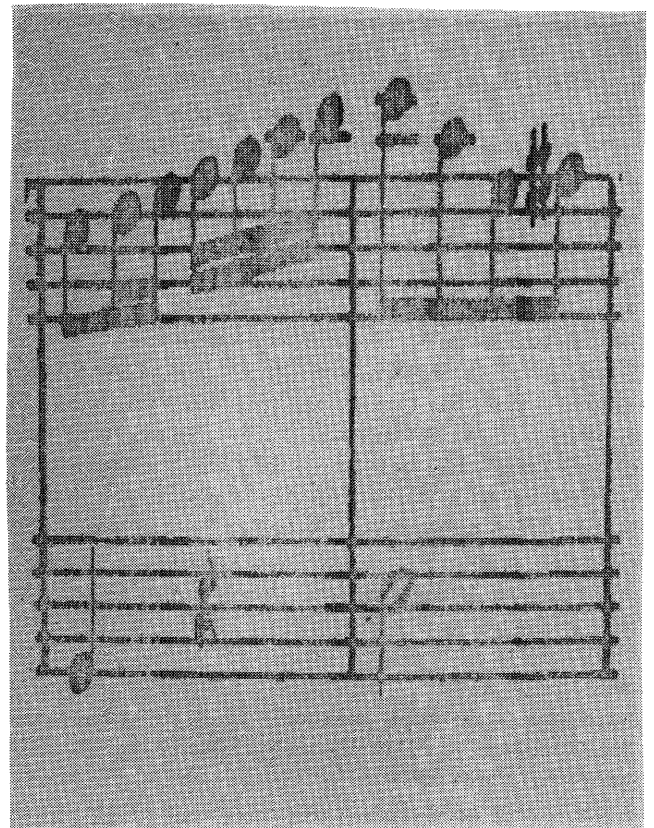


Figure 4—Printout of a picture

to one bit per point. This is done by choosing a threshold, θ , and considering all points with transmittance $T > \theta$ to be "White", and all points with transmittance $T < \theta$ to be "Black". Since the transmittance of most points in the picture will not be near the threshold, the choice of θ is not crucial.

The result is a matrix whose entries correspond to the digitized points of the music sample. (One such matrix is shown in Figure 4, representing "Black" by a point and "White" by a blank for display purposes.) Thresholding is done by a routine which is called PACK since it also packs the matrix for storage economy. This packed matrix is the data-base for the program.

ISOLATION

As shown in Figure 3, the packed Black-White matrix produced by the Input Section is passed to the Isolation Section. The function of this Section is to isolate each of the music notational symbols represented in the matrix, so that the program can then attempt to recognize individual symbols (rather than having to deal with arbitrary groups of symbols or with parts of symbols). The symbols must be isolated from the staff-lines upon which they are superimposed, and from each other. The staff-lines can be considered as graphical interference; they connect symbols that would normally be disconnected, they camouflage the contours of symbols, and they fill in symbol areas that would normally be blank. Thus, recognition is greatly simplified if the symbols are extracted from the staff-lines. This is a problem of pattern recognition in the presence of qualitatively defined interference, as the graphical positions of the staff-lines are qualitatively defined (i.e., five lines, nominally horizontal, straight, and equally spaced).

The process of symbol isolation must not destroy or significantly distort the original picture, for such destruction or distortion will make recognition difficult if not impossible. A method of isolation with minimal picture distortion has been developed. This method, called Fragmentation and Assemblage, is illustrated in Figure 5. First, the symbol fragments falling between staff-lines are picked out from the staff-line background by a relatively complex Fragmentation procedure. Each fragment is obtained by finding a list of the points that make up its contour (considering its intersections with staff-lines as part of its contour). Then, these fragments are assembled together again by associating the fragments into sets called picture components, each picture component corresponding to exactly one of the music notational symbols of the original sample. In essence, this procedure reforms the symbols, but without the

ORIGINAL PICTURE:



FRAGMENTATION:



ASSEMBLAGE:



Small Dots indicate Connection.

Figure 5—Fragmentation and Assemblage

staff-line interference. The reformed symbols are isolated from each other, as desired.

For each fragment, a SLIP-list is produced containing ROWMAX, ROWMIN, COLMAX and COLMIN—the extreme rows and columns occupied by the fragment. This information is readily found from the fragment's list of contour points, and defines a rectangle bounding the fragment. The SLIP-list also contains fragment interconnection data.

In addition, a SLIP-list is produced for each component, containing a listing (by number) of the fragments that make up the component. It is then easy to find the overall ROWMAX, ROWMIN, COLMAX and COLMIN of the component from the data in the fragment SLIP-lists. (For example, the COLMIN of the component is just the minimum of the COLMINs of the fragments.) In this way, a bounding rectangle is found for each component.

The Fragmentation and Assemblage method is somewhat intricate and will be discussed in more detail in a future paper. However, it is important here to note the data reduction that it accomplishes. As illustrated in Figure 6, a picture is originally stored as a huge (250,000 point) matrix. The fragment data for the picture is stored as a fairly long table of contour points

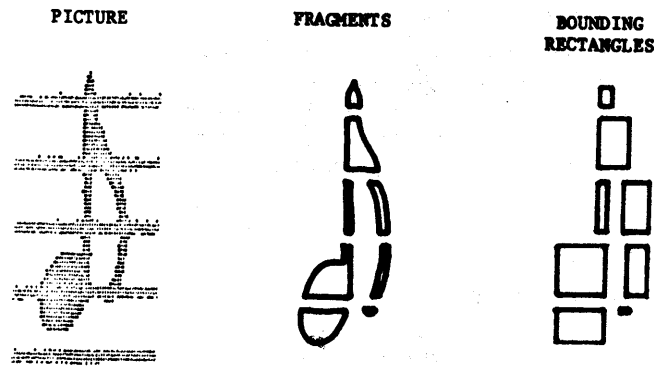


Figure 6—A picture, its fragments, and its bounding rectangles

(about 5000 to 10,000 contour points per picture). The corresponding SLIP-lists contain only the connection and bounding rectangle data (about 1000 list-entries per picture). In each step, the amount of data has been significantly reduced.

As will be seen, the small amount of easily-found information in the fragment and component SLIP-lists is enough, in most cases, to enable complete recognition of the components.

It was originally thought that each symbol would have to be fully reconstructed after Fragmentation by combining its fragments graphically, and filling in the spaces left by the staff-lines. This process was not needed. Surprisingly, the information in the Black-White picture matrix was never needed after Fragmentation. Even more surprisingly, the information in the fragment contour-point listings was needed only in five very special cases. Except for these special cases, all recognition can be done just from the ROWMAX, ROWMIN, COLMAX, COLMIN, and connection information in the SLIP-lists! This result points out another benefit of the Fragmentation-Assemblage method of symbol isolation: almost all the recognition tests can be performed on the relatively small data base of the SLIP-lists, rather than on the larger data base of the Fragment point-listings, or the even larger data base of the picture matrix.

PRELIMINARY FILTERING

The symbols of the picture having been isolated, the actual recognition procedure can begin. First, it is interesting to note that certain familiar techniques cannot be used. In standard music notation there are some symbols that are not characters, in that they have one or more graphical parameters whose value may be different for each occurrence of the symbol. Consider

Figure 7. On the top staff there are two occurrences of the same symbol. However, the two are not geometrically identical, since the spacing is affected by the notes on the bottom staff. Thus, these symbols are not characters. Many of the techniques used for recognition of characters, such as template-matching, cannot be used to recognize non-character symbols. Such techniques are therefore not very useful in the recognition of music notation. Alternative techniques must be employed.

Music notation has many strong syntactic properties. In combination with small lists of possibilities for each unknown symbol, these syntactic properties can be used with good results to recognize each component. Thus, it was decided to use a preliminary filter to reduce the set of possible symbols corresponding to the unknown to a small subset, and then to use the syntactic properties for unique classification.

Upon consideration of the form of the data after Fragmentation and Assemblage (i.e., the SLIP-lists) and of the graphical properties of music notational symbols, it was decided that a simple but powerful preliminary filtering could be obtained by examination of the normalized overall size of the component. A close look at the symbols of standard music notation reveals that each symbol type is significantly different in overall height or overall width from almost all others.

In order to find the size range of each music symbol, a survey of standard music notation was made. Many samples of each type of notational symbol were measured, and the overall height and width of each symbol tabulated. These measurements must be normalized since different samples of music will in general be of different scale. The average height of a staff-space (which shall be denoted as SPHGT) appears to be a good normalization factor, so that by dividing the physical dimensions by SPHGT, the measurements can



Figure 7—Non-character symbols in music notation

be converted into ratios that are independent of absolute dimensions.

Each normalized pair of measurements is plotted in a normalized-height vs. normalized-width property-space. A region is then delineated in the property-space for each type of music symbol. This region is chosen to be the smallest rectangular region enclosing all the plotted points for that symbol. The symbol regions in the property-space are shown in Figure 8. The figure shows, for example, that the measured natural signs were always approximately 0.6 to 0.8 SPHGTs in width, and 3.2 to 3.9 SPHGTs in height. Note from the figure that there is very little overlap of the regions.

In finding the list of possible assignments for an unknown component, it appears far better to include a few extra possibilities than to leave out the correct one. Therefore, it seems reasonable to enlarge the rectangular regions of the property-space to allow additional

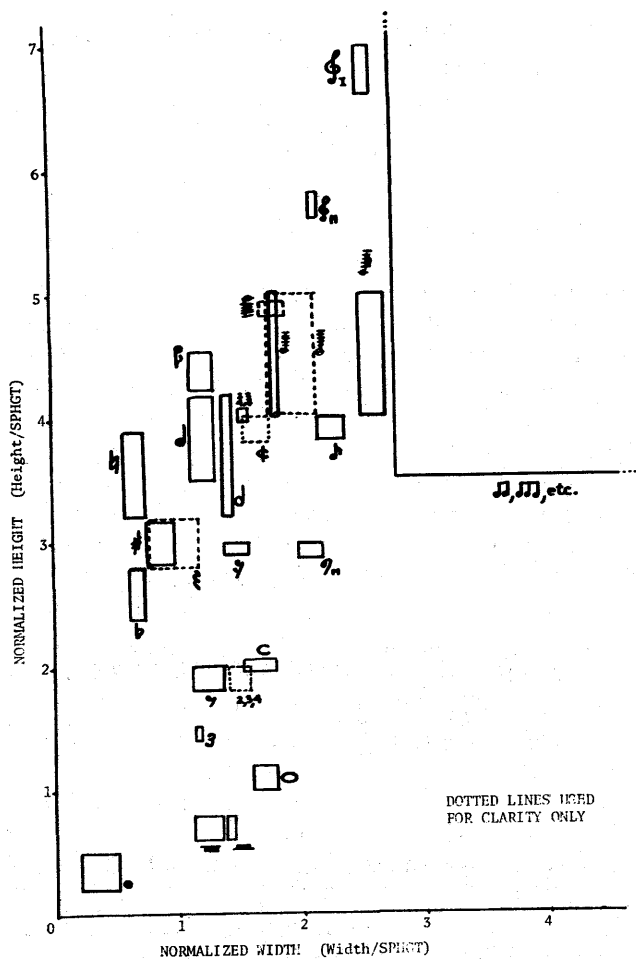


Figure 8—Normalized-height vs. normalized-width property-space

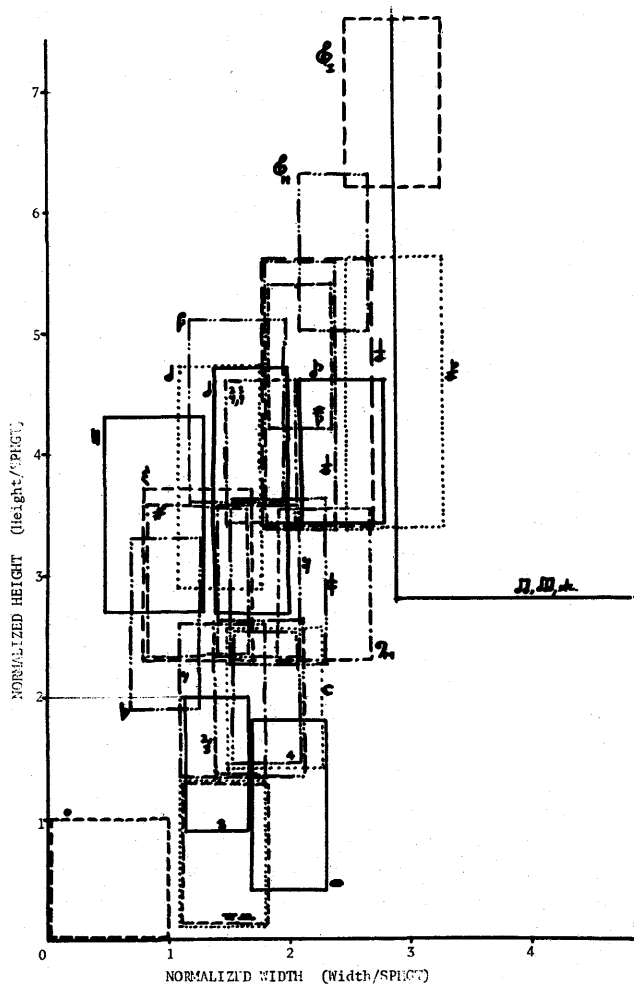


Figure 9—The H-W space (The normalized-height vs. normalized-width space with enlarged regions)

tolerance. This allows for processing effects due to the scanner and quantization, and for such characteristics of printed music as the widening of a symbol-line when it crosses a staff-line, etc. The normalized Height-Width property-space with the enlarged regions is called "the H-W space". This is shown in Figure 9. Note that the number of overlapping regions at a point is usually between three and five, with a maximum of eight. The points with the higher numbers of overlaps usually occur where the corners of many regions meet. Since most plotted points for unknown components will be found near the middle of the regions of their corresponding symbols, the areas of the property-space corresponding to the larger numbers of overlaps are rarely encountered in practice.

For every tested picture, the point in the H-W space plotted for each component fell in the region representing the symbol actually corresponding to that

component. Since the H-W space contains enlarged regions, and since it uses normalized values, it is reasonable to use the same H-W space for recognizing any sample of music notation. However, if for certain printed music it is found that the regions in the H-W space are not properly delineated, it is only necessary to find an H-W space based on this new set of symbols. Changing the H-W space in DO-RE-MI requires nothing more than changing a few parameter values. Or, one might store several H-W spaces, and then call the one corresponding to the music style in which the sample has been printed. This ability to change the H-W space so easily is an attribute of the modularity of the program.

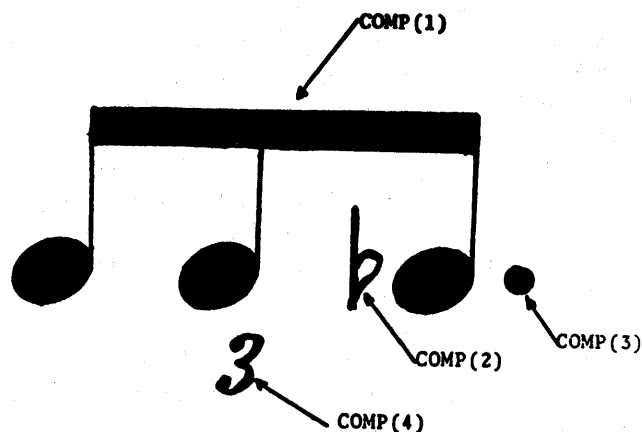
The Get-Possibilities Routine (GETPOS) performs the preliminary filtering. Given the H-W space, a short list can be generated of the symbols that can possibly correspond to a given unknown component. This is done by finding the point in the property-space representing the normalized-width and the normalized-height of the component. The regions in which this point falls specify the music symbols which can correspond to the component. When the Possibility List has been found by GETPOS, it is added to the SLIP-list of the component.

It is interesting to note the power of the preliminary filter. It is based on a simple overall property of the component, normalized size, and is completely independent of the internal features of the component. Yet, it is able to reduce the number of possible symbols corresponding to the component usually to about three to five with a maximum of eight. In addition, use of the data found in Fragmentation and Assemblage makes the determination of the normalized overall size a trivial calculation.

ORDERING

The components have been found by Fragmentation and Assemblage independent of their positions on the staves. It is therefore necessary to associate each component with the staff from which it was extracted, and to find the left-to-right order of the components within each staff. Knowledge of this ordering is needed to produce the final output sequence. In addition, it is useful because two important graphical features of symbols in music notation can be ascertained from the order information: immediate symbol context and symbol nestedness.

The Order Routine (ORDER) finds the left-to-right ordering of the components in each staff, forming an order list for each staff. Due to the two-dimensionality of the placement of music symbols, the ordering process is not simple. (This is in direct contrast to printed text



ORDERING BY LEFTMOST POINT:

COMP(1), COMP(4), COMP(2), COMP(3).

ORDERING BY CENTER OF MASS:

COMP(4), COMP(1), COMP(2), COMP(3).

ORDERING BY THE "ORDER" ROUTINE:

COMP(1)-L, COMP(4)-L, COMP(4)-R,
COMP(2)-L, COMP(2)-R, COMP(1)-R,
COMP(3)-L, COMP(3)-R.

Figure 10—An example of three different ordering methods

where the characters are in a one-dimensional string, and where the ordering process is trivial.) Consider the music sample of Figure 10. Note, for example, that Component 2 is neither to the left nor to the right of Component 1, but that a more complex two-dimensional relationship exists between them. Thus, ordering the components by one-dimensional techniques will not be satisfactory. An ordering by leftmost point would give the sequence: COMP(1), COMP(4), COMP(2), COMP(3). An ordering by center of mass would give: COMP(4), COMP(1), COMP(2), COMP(3). Neither is a good representation of the situation.

DO-RE-MI orders the components by both their leftmost and rightmost points in the same list. For Figure 10, this method gives the sequence: COMP(1)-Left, COMP(4)-Left, COMP(4)-Right, COMP(2)-Left, COMP(2)-Right, COMP(1)-Right, COMP(3)-Left, COMP(3)-Right. This ordering more accurately represents the overlapping of COMP(4) and COMP(2) by COMP(1).

SYNTACTIC TESTS

Most of the symbols of standard music notation (as opposed to the alphabetic symbols of printed text)

have a strong set of symbol-to-symbol contextual syntactic properties. For example, in the standard music considered by this study, the "flat" sign may appear in only one of two contexts (as shown in Figure 11):

- (1) In a key signature to the right of a clef or bar-line.
- (2) As an accidental to the left of a note.

Any other appearance of a flat is syntactically incorrect. Also illustrated in Figure 11 is the rule that the first symbol of a line of music must be a clef. This type of syntactic property can be used to great advantage in recognition.

Music notation contains many redundancies and these too can be used to aid recognition. There are two types of redundancies: syntactic redundancy and graphical redundancy. Figure 11 shows an example of syntactic redundancy. Since the line starts with a *G*-clef, the first flat of the key-signature (if any) must be on the middle line, i.e., a *B*-flat. Conversely, if the first symbol of the key-signature is on the middle line, it must be a flat. Continuing in the same manner, if the first key-signature symbol is a flat, the second key-signature symbol (if any) must be on the top space, and must be a flat. An example of graphical redundancy would be the *F*-clef, where a component cannot be

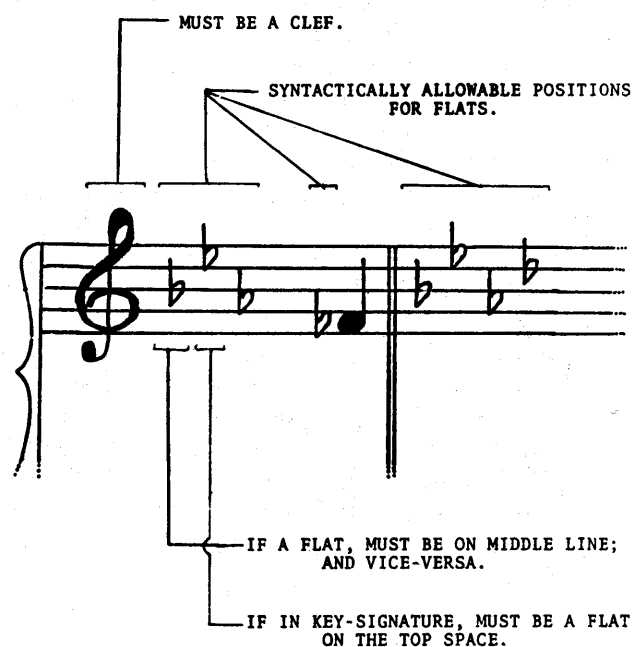


Figure 11—Syntax and redundancy in standard music notation

recognized as an *F*-clef, even if it passes all other tests, unless two other components are recognized as dots and are in the correct position to be the *F*-clef dots. (If so, all three components are considered to form the *F*-clef symbol).

Another type of syntactic property exhibited by music notational symbols is positional syntax. Many symbols can occur only in a fixed position on the staff, or only in a certain region (e.g., above the staff). For example, the two dots of a repeat sign will always be found in the two middle spaces of the staff. Also, each rest has a standard vertical position and thus will always occupy the same position on the staff (except for a few special cases).

The Syntax Routine (SYNTAX) uses syntactic properties to perform the final recognition of the components. It examines all the components in sequence as they appear on each staff's order list. For each component, SYNTAX performs tests on every entry on the Possibility List, and eliminates from the list all those possibilities which fail any of these tests. Contextual syntactic properties are tested. Some feature tests must be employed, but only when the other three types of tests still yield ambiguities. This occurs, for example, when a sharp, a flat or a natural appears as an accidental (i.e., to the left of a note). In this usage, the three symbols can be syntactically equivalent. Since they are approximately the same size and would occupy approximately the same vertical position, they often cannot be differentiated by the H-W space or by position tests. In this case, feature tests must be used. As will be seen, a simple feature test is used to differentiate among the three.

As shown in Figure 3, SYNTAX calls nine subroutines, each testing a different class of symbol-possibility, e.g., notes, rests, clefs, time-signatures, etc. SYNTAX2 is Part 2 of the Syntax tests, and is called when one of the five special cases requiring information from the Fragment contour-point lists is found. In all other cases, recognition is completed by SYNTAX and its nine subroutines, and the only information used is that in the SLIP-lists, i.e., the bounding rectangles on each fragment and component plus the interconnection data.

A REPRESENTATIVE 'SYNTAX' SUBROUTINE

As an illustration of the procedures used in SYNTAX and SYNTAX2, a representative SYNTAX subroutine, the Sofon Test, will be discussed in some detail. ("Sofon", pronounced sō-fon, is a term that is coined here from the initials "Sharp Or Flat Or Natural" to denote a symbol which is any of these three, inde-



Figure 12—Situations where sofons occur

pendent of whether the symbol appears in a key-signature or as an accidental. No such general term for this set of symbols exists in the music vocabulary, and such a term is needed since these symbols can often be treated together during symbol recognition.) The Sofon Test (SOFON) tests all components that have either “sharp” or “flat” or “natural” on their Possibility List.

There are strong contextual syntactic tests, redundancy tests and positional syntactic tests that can be applied to sofons. Thus, a sofon must be either:

1. To the left of a note or a beamed-together note-group, and close to it (i.e., within 1 SPHGT horizontally), and at least partially overlapping it vertically.
2. Overlapped horizontally by a beamed-together notegroup, and at least partially overlapping it vertically.

3. To the right of a clef or bar-line and on the correct pitch-space. (The “pitch-space” of a sofon is defined as the pitch of the line or space which the sofon is “on.” This is determined by the vertical position of the sofon and the clef currently in effect.) In this case, the correct pitch-spaces are:

For a sharp: *F*
 For a flat: *B*
 For a natural: *F* or *B*

4. To the right of another of the same sofon type, and close to it, and on the correct pitch-space. The sequence of correct pitch-spaces for key-signature sofons is as follows:

For sharps: *F, C, G, D, A, E, B*
 For flats: *B, E, A, D, G, C, F*
 For naturals: Either of the above sequences.

The initial sharp or flat of a key-signature can also be to the right of a natural.

In the first two cases the sofon is used as an accidental; in the latter two cases it is used in a key-signature. Typical situations are illustrated in Figure 12.

Since accidental sofons are syntactically equivalent, the three types of sofons often cannot be separated from each other by the above tests (though SYNTAX will eliminate all other possibilities from the Possibility List). In this case, a feature test must be used.

After consideration of many other separation

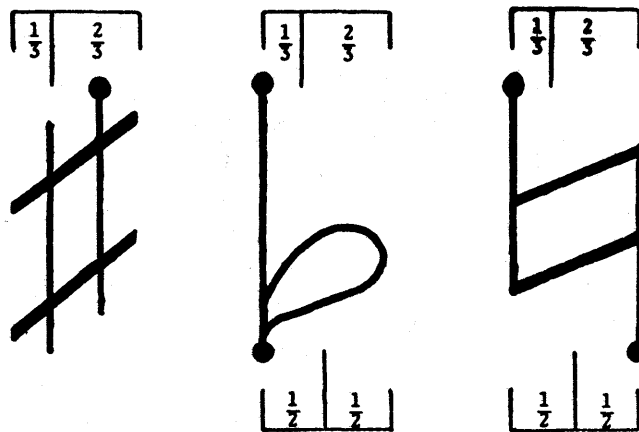


Figure 13—Sofon separation

algorithms, it was noted that the three sofos could be separated by examining only their top and bottom points, as follows (see Figure 13):

(1) Sharp vs. Flat; Sharp vs. Natural:

- For a sharp, the topmost point occurs in the *rightmost two-thirds* of the component width.
- For a flat or natural, the topmost point occurs in the *leftmost one-third* of the component width.

(2) Flat vs. Natural

- For a flat, the bottommost point occurs in the *left half* of the component width.
- For a natural, the bottommost point occurs in the *right half* of the component width.

Thus, examination of the topmost point of the component separates sharps from flats and naturals, and examination of the bottommost point separates flats from naturals.

The sofon separation test requires finding the topmost and bottommost points of the component, and these can be found easily from the contour-point lists. This is one of the cases where it may be necessary to employ SYNTAX2 and the fragment contour-point lists. Point lists for only two fragments must be examined: the fragments containing the ROWMIN and the ROWMAX of the component. In fact, often these fragments fall completely on one side or the other of the two-thirds or one-half points of the component's width. When this is so, the sofon separation can be made solely from the information in the SLIP-lists. Then, there is no need to look at the contour-point lists, and SYNTAX2 does not have to be called at all.

OUTPUT

The Output Routine (OUTPUT) takes the music which was recognized by the Syntax Routine (SYNTAX and SYNTAX2) and produces the final DO-RE-MI output according to the Ford-Columbia Music Representation. As has been mentioned, Ford-Columbia is an alphanumeric language which is substantially isomorphic to standard music notation. It was developed by a musician, Stefan Bauer-Mengelberg.⁵ OUTPUT stores the Ford-Columbia representation of the recognized music on a SLIP list. For example, a printout of the final output for the picture of Figure 4 is shown in Figure 14. (Output restrictions on SLIP-lists in CTSS

```
(('I O 1,(26 E,(27 S, 28 S)),((29 S, 30 S, 31 S, 32 S)),/,(33 E, 31 E,
28 E, 29=E), 'I O 2, 21 O, R O,/ 26 H)
```

Figure 14—Output for the picture of Figure 4

do not allow accurate printing of all the symbols of the Ford-Columbia character set; therefore, a modified Ford-Columbia is printed out with commas representing blanks and apostrophes representing exclamation points.) In this printout, "I01" indicates that the first instrument's line begins, "26E" indicates a *Eighth* note on space 26 (the third space on the top staff), parentheses indicate beams, "/" indicates a bar-line, etc.

RESULTS OF TEST RUNS

DO-RE-MI was tested on some representative pictures, including the one shown in Figure 4. In all cases, the program produced the desired Ford-Columbia representation of the input picture with complete accuracy. All symbols in the subset of music notation considered by DO-RE-MI were correctly recognized. In addition, all symbols not in that subset were correctly recognized as such.

Some overall statistics on the test-runs: DO-RE-MI correctly isolated all the music notational symbols into a total of 137 components. These components were formed by 527 fragments. All 137 components were correctly recognized by DO-RE-MI (including thirteen components which were each a group of three to four notes beamed together). An average test-run took about four minutes, from packed Black-White matrix to Ford-Columbia output, for a test picture of two to three measures of duet music (i.e., four to six single measures of music). This time figure could be significantly reduced by various means; however, minimization of run-time was not a major goal of this work.

CONCLUSION

This work is an investigation into computer recognition of a class of conventionalized, two-dimensional, visual patterns: standard engraved music notation. Important aspects of the problem involve pattern recognition in qualitatively-defined interference, recognition of positionally two-dimensional patterns, use of syntax and redundancy properties in recognition, and recognition of non-character symbols. A simple preliminary filter

proved very effective. Bounding rectangles on fragments and components unexpectedly provided all necessary data for recognition in almost all cases, and thus examination of detailed feature properties was rarely required. The program produced should be able to be expanded to the recognition of all printed music. In addition, the pattern recognition techniques developed can possibly be applied to computer recognition of such things as maps, graphs, organic chemistry symbols, circuit diagrams, blueprints, aerial photographs, etc.

ACKNOWLEDGMENT

I would like to gratefully acknowledge the encouragement, support, and counsel of Professor Murray Eden of M.I.T., who guided me throughout the course of this research, and of Professor Allen Forte of Yale, Dr. Oleh Tretiak of M.I.T. and Professor Francis Lee of M.I.T. The interest and assistance of the members of the Cognitive Information Processing Group of the

M.I.T. Research Laboratory of Electronics is also greatly appreciated.

REFERENCES

- 1 M EDEN
Recognizing patterns
P Kolars and M Eden editors Chapter 8
The MIT Press Cambridge MA 1968
- 2 H B LINCOLN
The current state of music research and the computer
Computers and the Humanities September 1970
- 3 D H PRUSLIN
Automatic recognition of sheet music
Doctoral Thesis MIT Cambridge MA January 1967
- 4 O J TRETIAK
Scanner display (SCAD)
Quarterly Progress Report, No. 83 MIT Research
Laboratory of Electronics October 1966
- 5 S BAUER-MENDELBERG
(of the IBM Systems Research Institute New York City)
*Representing music to a computer: A primer of the
Ford-Columbia music representation (DARMS)*
Manuscript in preparation

A storage cell reduction technique for ROS design

by Dr. C. K. TANG

International Business Machines Corporation
Endicott, New York

INTRODUCTION

A Read Only Store (ROS) of n inputs (n -bit address) and m outputs (m -bit words) stores $2^n \times m$ bits of information, and is abbreviated as $(2^n \times m)$ -bit ROS. The use of monolithic arrays as a ROS depends upon an effective store (write once) procedure (or personalization of the ROS). Two procedures are currently being considered: the personalization by final metallization pattern (masking),¹ and the personalization by post-metallization connection elimination technique (etching or zapping).^{1,2} This study describes a method of designing a monolithic ROS which will reduce the number of storage cells required in the former method of personalization.

The ROS personalization pattern, which is usually described by a truth table of 2^n rows (where n is the number of variables of the function to be implemented by the ROS), is first expressed as a function completely expanded with respect to all of its n variables, i.e., a function in the standard sum form. The incomplete expansion of the same function with respect to less than n variables is used to show that the number of storage cells in the ROS can be reduced from 2^n to either 2^{n-1} , or 2^{n-2} , or 2^{n-3} , etc.; this is made possible by using a few additional logic gates in the ROS to implement a set of selected simple functions (called functional sets). As an extension of the previous method of simple expansion, a double expansion method is also presented; this enables the expansion of the function with respect to a smaller number of variables to be practical.

THE USE OF A FUNCTION SET

A ROS can be represented in block diagram form as shown in Figure 1. The inputs cause only one of the 2^n decode outputs to be energized; the energized decode output selects the m storage bits to the output through the sense amplifier.

If one takes a logic viewpoint of the ROS, then $(2^n \times m)$ -bit ROS can be expressed as m logic functions of n input variables, and each of the m logic functions corresponds to one of the m outputs of the ROS. Let i_1, \dots, i_n be the n inputs to the ROS, then any one of the m outputs can be expressed as a logic function F of the n inputs, and F can always be expanded with respect to all the input variables as follows:

$$\begin{aligned} F &= F(i_1, \dots, i_n) \\ &= \bar{i}_1 \bar{i}_2 \dots \bar{i}_n F(0, 0, \dots, 0) + \bar{i}_1 \bar{i}_2 \dots i_n F(0, 0, \dots, 1) \\ &\quad + \dots + i_1 i_2 \dots i_n F(1, 1, \dots, 1) \end{aligned} \quad (1)$$

where \bar{i}_1 denotes the negation of i_1 .

Each of the 2^n products of the form $i_1^* i_2^* \dots i_n^*$ in Equation (1) (where i^* represents either i or \bar{i}) is one of the decode outputs, and hence the corresponding residue function denotes the stored bit (0 or 1) associated with that decode output. If personalization by masking is used, the storage of the information bit 1 or 0 is usually done by the connection or disconnection of a cell in the ROS (where the cell is usually a transistor or the emitter of a multiple-emitter transistor). Hence, there are 2^n cells required for each output bit of the ROS. These cells are OR gated to provide the desired function F .

Now, if the function F is expanded with respect to $n-1$ of its n inputs, then there are only 2^{n-1} product terms; i.e.,

$$\begin{aligned} F &= F(i_1, \dots, i_n) \\ &= \bar{i}_1 \bar{i}_2 \dots \bar{i}_{n-1} F(0, 0, \dots, 0, i_n) \\ &\quad + \bar{i}_1 \bar{i}_2 \dots i_{n-1} F(0, 0, \dots, 1, i_n) + \dots \\ &\quad + i_1 i_2 \dots i_{n-1} F(1, 1, \dots, 1, i_n), \end{aligned} \quad (2)$$

Let $V_1(i_n)$ be the set of all functions of the single variable i_n . The outputs of $V_1(i_n)$ will be $(0, 1, i_n, \bar{i}_n)$, and $V_1(i_n)$ will be called *the function set of the single variable i_n* . Each of the residue functions in Equation (2) is a

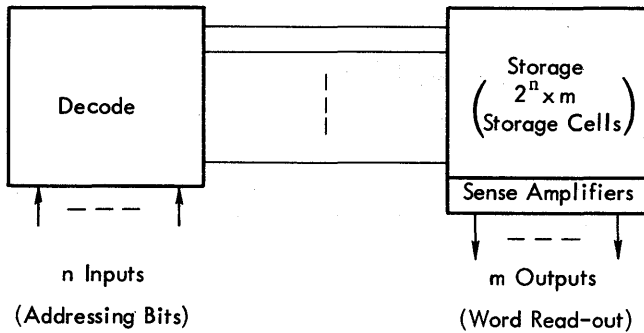


Figure 1— $(2^n \times m)$ -bit ROS

function of the single variable i_n , and hence is included in the function set $V_1(i_n)$. Assume the function set $V_1(i_n)$ and the decode with i_1, \dots, i_{n-1} as its inputs are available, then one can easily see from Equation (2) that if there exist 2^{n-1} cells and each is capable of performing a two-way AND function, then by OR gating these 2^{n-1} cells, the arbitrary function $F(i_1, \dots, i_n)$ is obtained.

The above statement suggests the construction of a $(2^n \times 1)$ -bit ROS. The selection of a particular function from the four functions of $V_1(i_n)$ for each cell is done by making a connection in the final masking from each cell to an appropriate line that carries the desired function (note there are four lines carrying 0, 1, x_n , \bar{x}_n). By sharing the four lines carrying $V_1(i_n)$ and the $(n-1)$ -input decode, a $(2^n \times m)$ -bit ROS can be constructed by using $2^{n-1} \times m$ storage cells. This arrangement is shown in the block diagram of Figure 2. Note that the associated decode circuits should also be simpler since an $(n-1)$ -input decode is required instead of an n -input decode. Also, note that the function set $V_1(i_n)$ does not take any extra circuits to build because 0 and 1 stand for the ground and the power supply voltage, respectively, and both i_n and \bar{i}_n are necessary in building the n -input decode in a conventional ROS. In some conventional ROS designs the decoding of the n input variables may not be as explicit as that indicated by Figure 1, but the idea of using the V_1 function set can still be applied with a resultant saving of storage cells. This will be illustrated for the design of a current switch emitter follower ROS and a T^2L ROS in Appendix A and Appendix B, respectively. The existence of the simple cell that can perform a two-way AND is also illustrated in the appendices.

The use of the function set of a single variable can be extended to the function of more than one variable. For example, let $V_2(i_{n-1}, i_n)$ be the function set of two

variables i_{n-1} and i_n , defined as the set of all functions of the two variables i_{n-1} and i_n . There are 16 functions of two variables. The expansion of the function F with respect to the first $n-2$ inputs is given as follows:

$$\begin{aligned}
 F &= F(i_1, \dots, i_n) \\
 &= \bar{i}_1 \bar{i}_2 \dots \bar{i}_{n-2} F(0, 0, \dots, 0, i_{n-1}, i_n) \\
 &\quad + \bar{i}_1 \bar{i}_2 \dots i_{n-2} F(0, 0, \dots, 1, i_{n-1}, i_n) \\
 &\quad + \dots + i_1 i_2 \dots i_{n-2} F(1, 1, \dots, 1, i_{n-1}, i_n) \quad (3)
 \end{aligned}$$

Each of the residue functions is a function of two variables, i_{n-1} and i_n , and is obtainable from the function set $V_2(i_{n-1}, i_n)$. There are only 2^{n-2} products in Equation (3) and so now only 2^{n-2} storage cells are required. Each cell should also be able to perform the two-way AND function and can be OR gated.

Obviously the use of the function set can be extended to any number of variables k , where $k < n$. When $k=3$, V_3 will have $2^3=256$ functions. Although the number of storage cells required (2^{n-3}) is only one-eighth of the conventional ROS, the circuits required to provide the 256 signals and the wires required to carry these signals for selection may now outweigh the saving on storage cells for small ROS. For large ROS (e.g., $2^{12} \times 32$ -bit ROS) this arrangement may be practical. In such a design, if 1024 storage cells are to be implemented in one chip, 16 chips are required for the $2^{12} \times 32$ ROS, and it becomes more suitable to implement the circuits that produce half of the 256 functions (no two of these functions are complementary to each other) in one separate chip and to place 128 input terminals on each storage chip to receive these 128 functions. Only emitter followers and inverters are required in each storage chip to generate all 256 functions required. By similar

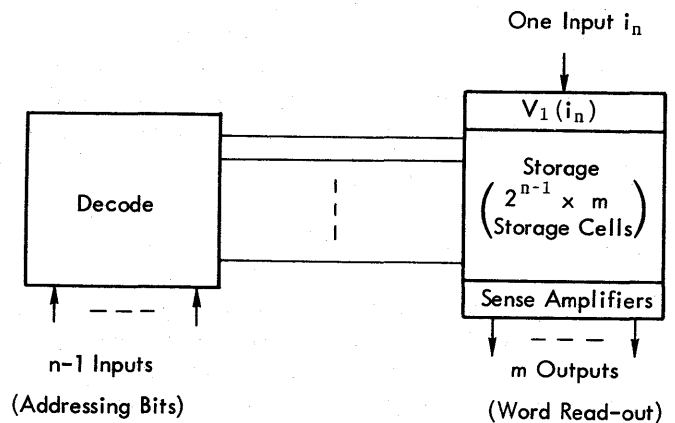


Figure 2— $(2^n \times m)$ -bit ROS using $2^{n-1} \times m$ storage cells

reasoning, much less than 128 input terminals can be placed on each storage chip to reduce the I/O pin requirement of the chip, and all the 256 functions will be generated inside each chip by some simple logic circuits.

This idea of producing some of the functions of V_3 in a separate chip to reduce the circuit requirement of each chip can be applied equally well to ROS design using the V_2 functional set. The function set of V_4 has $2^4 = 65536$ functions and it becomes impractical to construct a ROS by using only 2^{n-4} storage cells. However, the method described in the next section overcomes this difficulty.

DOUBLE EXPANSIONS

Expand the function F in the following way:

$$\begin{aligned}
 & F(i_1, \dots, i_n) \\
 &= \bar{i}_1 \bar{i}_2 \dots \bar{i}_{n-4} F(\underbrace{0, 0, \dots, 0}_{n-4}, i_{n-3}, i_{n-2}, i_{n-1}, i_n) \\
 &+ \bar{i}_1 \bar{i}_2 \dots i_{n-4} F(0, 0, \dots, 1, i_{n-3}, i_{n-2}, i_{n-1}, i_n) \\
 &+ \dots + i_1 i_2 \dots i_{n-4} F(1, 1, \dots, 1, i_{n-3}, i_{n-2}, i_{n-1}, i_n)
 \end{aligned} \quad (4)$$

Each residue is a function of four variables and can be further expanded with respect to two of its four variables. For example, the residue function in the first row of Equation (4) is expanded as follows:

$$\begin{aligned}
 & F(\underbrace{0, 0, \dots, 0}_{n-4}, i_{n-3}, i_{n-2}, i_{n-1}, i_n) \\
 &= [i_{n-3} + i_{n-2} + F(\underbrace{0, 0, \dots, 0}_{n-4}, 0, 0, 0, i_{n-1}, i_n)] \\
 &\quad \cdot [i_{n-3} + \bar{i}_{n-2} + F(0, 0, \dots, 0, 0, 1, i_{n-1}, i_n)] \\
 &\quad \cdot [\bar{i}_{n-3} + i_{n-2} + F(0, 0, \dots, 0, 1, 0, i_{n-1}, i_n)] \\
 &\quad \cdot [\bar{i}_{n-3} + \bar{i}_{n-2} + F(0, 0, \dots, 0, 1, 1, i_{n-1}, i_n)] \quad (5)
 \end{aligned}$$

Note that the second level expansion given by Equation (5) is in the product of the sum form while the first level expansion given by Equation (4) is in the sum of the product form.

Let $i_{n-3} + i_{n-2} + V_2(i_{n-1}, i_n)$ denote the set of functions obtained by OR gating each function in $V_2(i_{n-1}, i_n)$ with $(i_{n-3} + i_{n-2})$. There are 16 functions in this new function set $[i_{n-3} + i_{n-2} + V_2(i_{n-1}, i_n)]$ since there are 16 functions in $V_2(i_{n-1}, i_n)$. Function sets $i_{n-3} + \bar{i}_{n-2} + V_2(i_{n-1}, i_n)$, $\bar{i}_{n-3} + i_{n-2} + V_2(i_{n-1}, i_n)$ and $\bar{i}_{n-3} + \bar{i}_{n-2} + V_2(i_{n-1}, i_n)$ are similarly defined. These

four function sets represent a total of 64 functions. If we assume that half of the V_2 functions are available as an input to the ROS module (only five addition input terminals are required) as described in the last section, then some simple circuits in the storage chip can generate all the 64 functions of the four function sets. Also assume that the $(n-4)$ -input decode is available to the storage chip; then from Equations (4) and (5) it is clear that a storage chip of 2^{n-4} storage cells can produce the function F , if each of the cells can perform a five-way AND function. For example, to obtain the first row of Equation (4), a cell is used to AND gate the decode output $\bar{i}_1 \bar{i}_2 \dots \bar{i}_{n-4}$ and the four rows of Equation (5).

Each row of Equation (5) is a function in one of the four function sets previously described. The personalization is made by making the connections in final masking from the inputs of the cells to the lines that provide the functions.

For the ROS previously described, although for each output bit only 2^{n-4} storage cells are required as compared to the 2^n required by the conventional design, the reduction in circuitry is not by a factor of 16 due to the increased complexity of the five-way AND gate storage cells. A realistic estimate of reduction in circuitry can be made by calculating the silicon area and power dissipation of the circuits given in Appendices A and B.

In this section, the function is first expanded with respect to two variables, and then the residue functions are expanded with respect to another two variables. Combination of different numbers of variables being expanded are possible, as will be illustrated in Appendix B. The second level expansion given by Equation (5) takes the product-of-sums form. The second level expansion can also take the sum-of-products form, which will find its application in the current switch design, given in Appendix A. Such an expansion is given as follows:

$$\begin{aligned}
 & F(\underbrace{0, 0, \dots, 0}_{n-4}, i_{n-3}, i_{n-2}, i_{n-1}, i_n) \\
 &= [\bar{i}_{n-3} \bar{i}_{n-2} F(\underbrace{0, 0, \dots, 0}_{n-4}, 0, 0, 0, i_{n-1}, i_n)] \\
 &\quad + [\bar{i}_{n-3} i_{n-2} F(0, 0, \dots, 0, 0, 1, i_{n-1}, i_n)] \\
 &\quad + [i_{n-3} \bar{i}_{n-2} F(0, 0, \dots, 0, 1, 0, i_{n-1}, i_n)] \\
 &\quad + [i_{n-3} i_{n-2} F(0, 0, \dots, 0, 1, 1, i_{n-1}, i_n)] \quad (5')
 \end{aligned}$$

The corresponding four function sets are $\bar{i}_{n-3} \bar{i}_{n-2} V_2$, $\bar{i}_{n-3} i_{n-2} V_2$, $i_{n-3} \bar{i}_{n-2} V_2$ and $i_{n-3} i_{n-2} V_2$.

CONCLUSION

A function set can be used to greatly reduce the number of storage cells and the associated decode circuits in the ROS design. The cost trade-off is the additional circuits to generate the function set and the increased wiring complexity in "personalization" which is done by connecting each cell to the appropriate function set output lines by final metallization. For ROS with a large number of output functions (for example, consider a microprogram store application: the number of input variables (addressing bits) is typically from 10 to 15, and the number of output functions (bit lines) is typically from 32 to 128), the cost of circuits to generate the function set will be relatively small since the function set can be shared by all output functions. Since the storage cells in a ROS represent the major cost of a ROS, the storage cell reduction technique described in this paper may produce a severalfold cost reduction in the ROS manufacture.

ACKNOWLEDGMENT

The author wishes to thank his colleague, Dr. G. G. Langdon, Jr., for his encouragement and suggestions.

REFERENCES

- 1 A BERGH J C BARRETT J E PRICE
Design considerations for a high-speed bipolar read-only memory
1970 IEEE International Solid-State Circuits Conference
- 2 *Mass-produced read-only memory is customwired after assembly*
Electronics August 18 1969
- 3 M H LEWIN
A survey of read-only memories
Proceedings of the 1965 Fall Joint Computer Conference
- 4 R A HENLE I T HO G A MALEY
R WAXMAN
Structure logic
Proceedings of the 1969 Fall Joint Computer Conference

APPENDIX A

CURRENT SWITCH EMITTER FOLLOWER ROS DESIGN

For comparison, a conventional current switch emitter follower ROS design⁴ of 256×1-bit is shown in

Figure A-1. A current switch emitter follower ROS of the same number of bits using V_2 function set described in the Introduction is shown in Figure A-2. The * at the base of each transistor denotes the connection to the appropriate line of $V_2(i_7, i_8)$ function set (or connection to one of the 15 lines, and open base means "0" function in V_2 function set). The collector of each storage transistor is connected to a vertical decode line. Note that only one transistor will be turned on at a time in each column. Therefore no heavy drain of current will occur on any vertical decode line at any time. Only 64 storage transistors are required as compared to 256 storage transistors in Figure A-1. Figure A-3 gives a different design where the collectors of the storage transistors are fixed to ground, but 64 emitter decode outputs are required. This arrangement may be suitable in a large ROS where these 64 decode signals can be shared by other outputs of the ROS, and the storage and output circuit of Figure A-3 is only a small part of the large multiple output ROS.

Figure A-4 gives the design of a current switch ROS using the double expansion given by Equations (4)

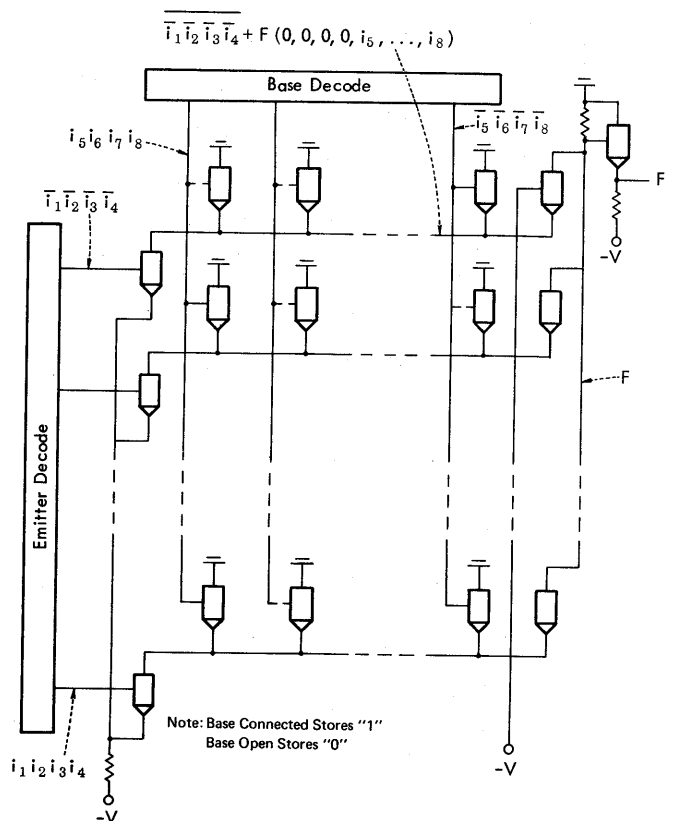


Figure A-1—Conventional current switch emitter follower ROS

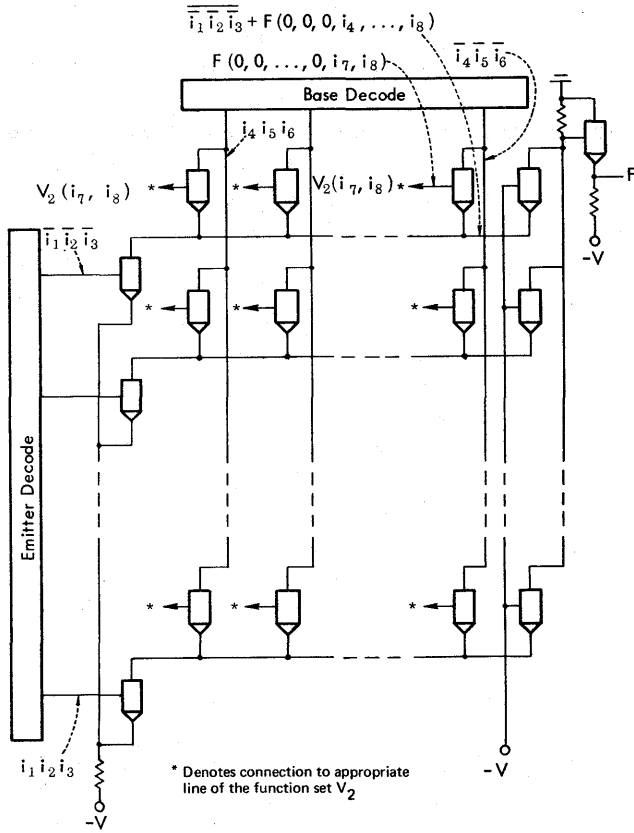


Figure A-2—Current switch emitter follower ROS using V_2 function set

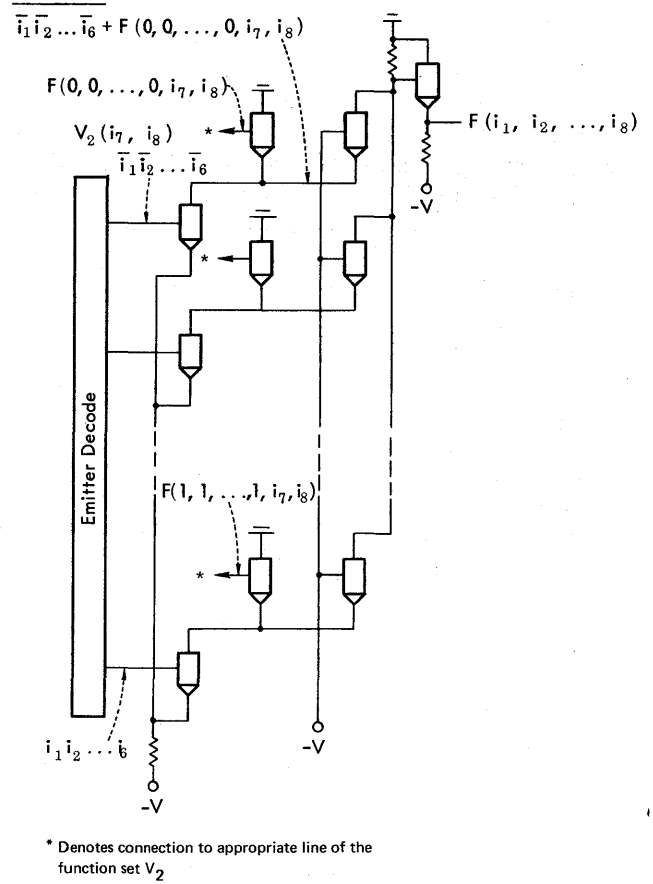


Figure A-3—Current switch emitter follower ROS with grounded collector using V_2 function set

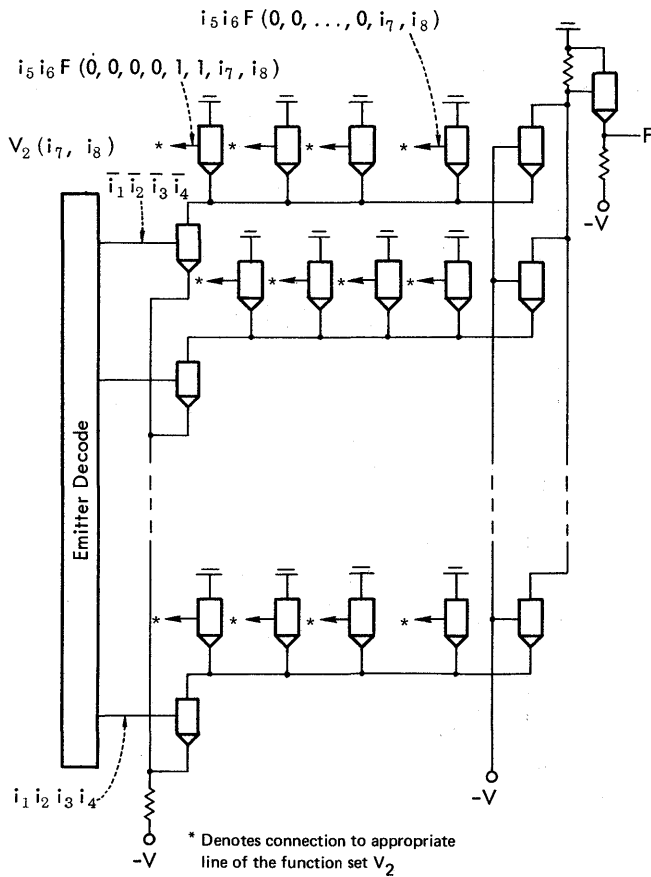


Figure A-4—Current switch emitter follower ROS using the double expansions

and (5') in the Double Expansion section. Sixty-four signal lines carrying the four function sets are required. Like the design of Figure A-3, it will be suitable for large ROS.

APPENDIX B

T²L ROS DESIGN

For comparison, a conventional T²L ROS design of 256×1-bit is shown in Figure B-1. A T²L ROS of the same number of bits using $V_2(i_7, i_8)$ function set described in the section entitled The Use of a Function Set is shown in Figure B-2. The * at the emitter of the multiple emitter transistors denotes again the connection to the appropriate line of $V_2(i_7, i_8)$ function set. Figure B-3 gives the design of a 256-bit T²L ROS using

the double expansions given by Equations (4) and (5) of the Double Expansion section. There are four function sets which require a total of 64 signal carrying lines. The circuits that generate these signals and the decode signals are not shown here. These circuits are supposed to be shared by other chips in a large multiple output ROS.

To reduce the number of signal lines for function sets, double expansions can be used with one more variable expanded in Equation (5). Figure B-4 illustrates such design. Only 32 lines carrying function set signals are required. Note that the circuit configuration is identical to that of the conventional design of Figure B-1, while the number of emitters in each multiple emitter transistor is reduced from 20 to 9.

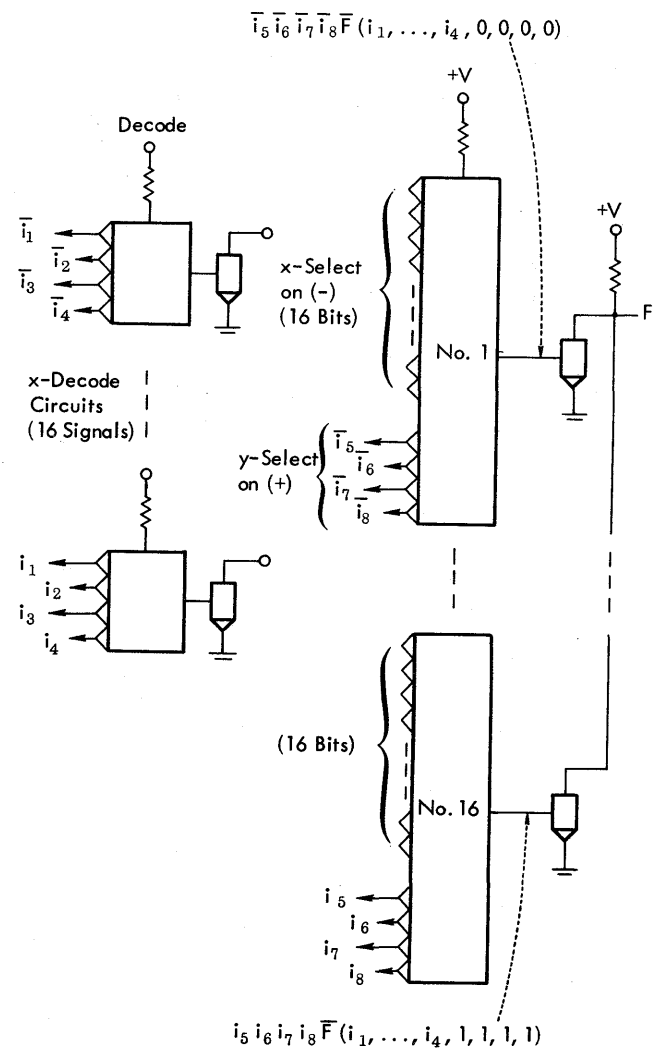


Figure B-1—Conventional T²L ROS

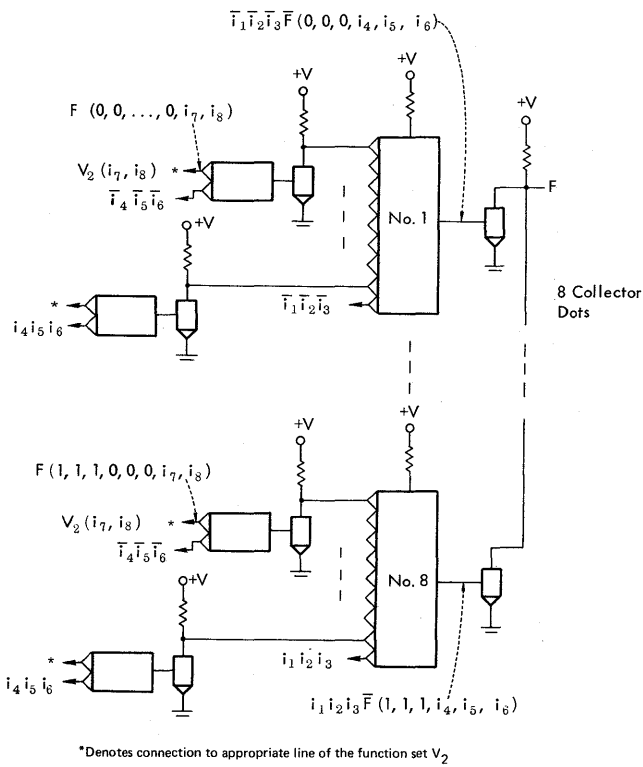


Figure B-2— T^2L ROS using V_2 function set

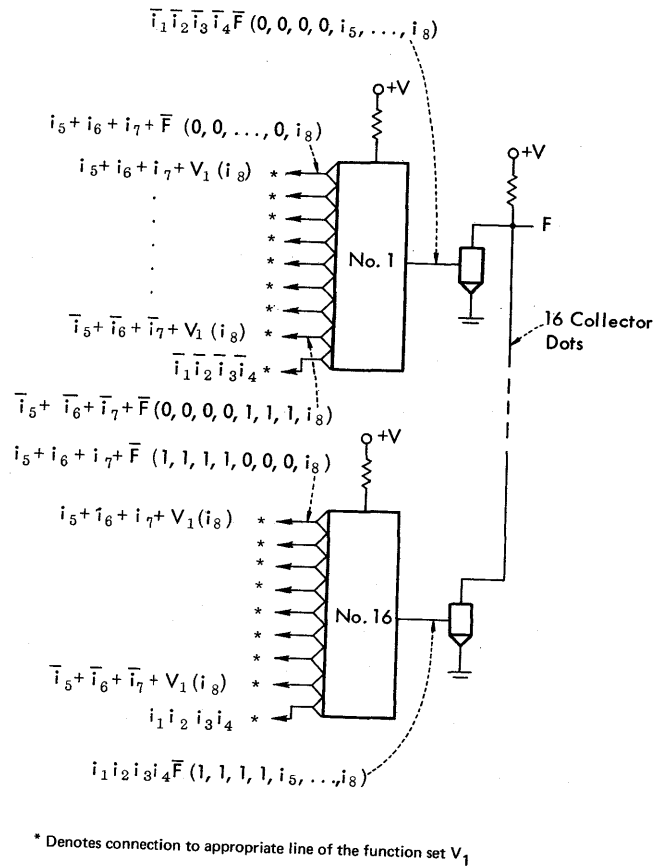


Figure B-4— T^2L ROS using the double expansion with one more variable expanded

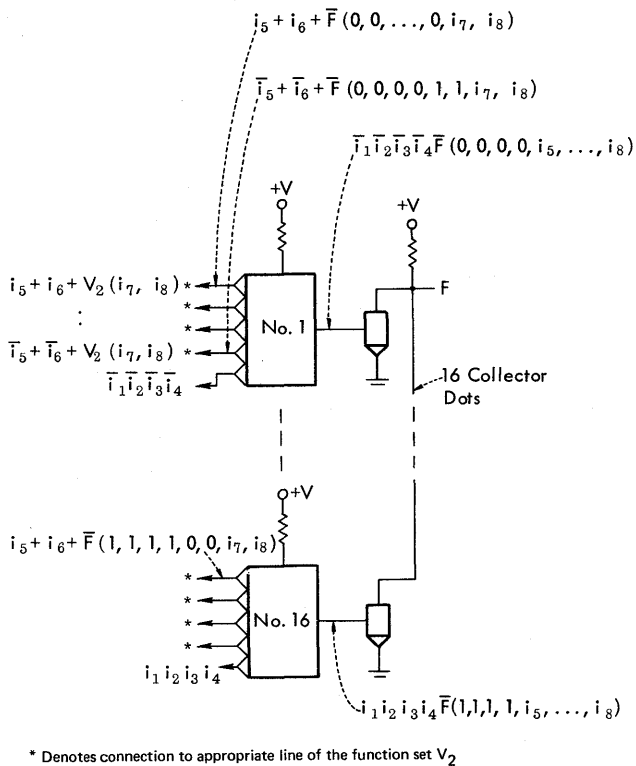


Figure B-3— T^2L ROS using the double expansions

A new approach to implementing high-density shift registers

by TEH-SEN JEN

IBM Components Division
Hopewell Junction, New York

INTRODUCTION

Progress in the last several years has made many LSI products practical for edp applications. One such product is in the area of FET dynamic shift registers. Through the use of novel circuit techniques and device processes,¹⁻³ the density, yield, and performance have all been increased appreciably. As a result, the cost has been so reduced that the use of FET dynamic shift registers for digital information processing has become increasingly popular and important. As cost is mostly dictated by density and yield, the author studied ways to improve these factors, in addition to the direct approach of designing better circuits and processes. This paper describes use of the logic organization of shift registers to achieve density and yield improvements. First, the logic organization of current shift registers is briefly reviewed, and then a new approach is presented. Although it is general, the discussion will be centered on integrated FET dynamic shift registers because of their practical importance. Implications and trade-offs of using the new approach on the device and circuit techniques will also be discussed.

LOGIC ORGANIZATION OF CONVENTIONAL FET DYNAMIC SHIFT REGISTERS

A basic shift register cell, in principle, requires a gated register circuit and a delay element.⁹ In practice, however, the delay element is almost always replaced by another gated register circuit, and thus a one-digit

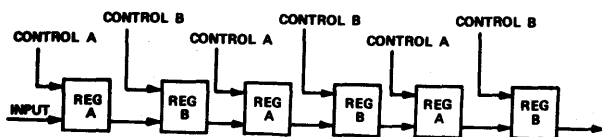


Figure 1a—Logic organization of conventional shift registers

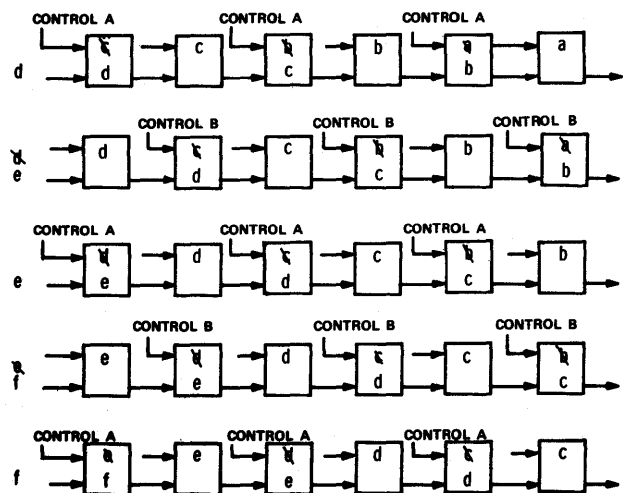


Figure 1b—Data flow of conventional shift registers

shift register cell actually consists of two gated registers. To store N digits of data, N cells or $2N$ registers are required. Figure 1a shows such a shift register. The shift operation is performed in the following way: The A control gates the output of register B (or the input) into register A while the output of register B (or the input) remains unchanged during the operation. The control B then gates the output of register A into register B while the output of register A remains constant. By successive applications of control A and control B , the data are shifted from the left to the right, as demonstrated in Figure 1b.

Basically, all the FET dynamic shift registers are operated in this way. Every shift register cell basically has two stages, and each stage functionally is a gated register circuit. The "register" in this case is simply a capacitor, either a parasitic or an enhanced parasitic one. The control of the gated register circuit may be one signal or one set of clock signals to perform gating, resetting or setting functions.

THE MULTICONTROL ORGANIZATION

Note that in Figure 1b, every digit of information is always stored redundantly in two register circuits except during the transient of shift operations. For the sole purpose of storing information (without shifting), one register circuit alone is sufficient to store one digit of information. In shift registers, the additional register circuit for every digit position is to provide reliable shifting operations. However, this is true only if every digit of data is shifted at every control clock. In other words, two register circuits per shift cell permits all the data to be shifted simultaneously or "in parallel" for every shift operation. The situation is different if the shift operation is "serialized." That is, to shift only a portion of the stored data at one time, only the data undergoing shifting need two register circuits per digit, while the data not undergoing shifting at that moment do not need the additional register circuits. To illustrate this point, the shift register is reorganized into the form shown in Figure 2a. Instead of shifting all the data in the shift register at one time, the controls are sequenced so that only one of every three digits is shifted at one time. In effect, the shift operation is serialized. When a shift operation is to be performed, control *D* first gates the information from register *C* into register *D*. (Redundant information is stored in register *C* and register *D* after this operation.) Control *C* then gates the information from register *B* into register *C*. After this is finished, control *B* gates the information from register *A* into register *B*, and then control *A* shifts the data from register *D* into register *A*. The details of the operation are shown in Figure 2b.

It is clear that shift registers organized this way have a smaller maximum data rate, because of the serialized shift operation. This shortcoming can be removed, however, and will be discussed in the next section. A comparison of the conventional organization and the one illustrated in Figure 2a shows the following differences. The conventional one needs two register circuits for one digit of information, and two (or two sets of) controls. To shift *N* digits of information, *2N* circuits are needed, and every control needs to drive *N* loads. On the other hand, the one shown in Figure 2a needs

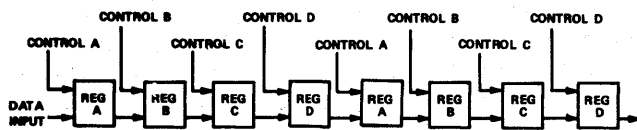


Figure 2a—Logic organization of *M*-control (*M*=4) shift register

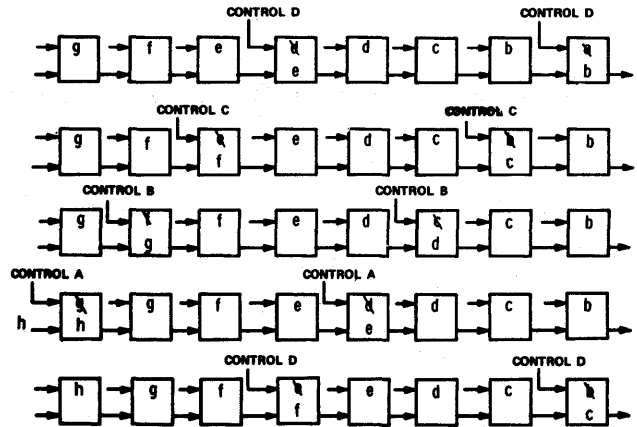


Figure 2b—Data flow of *M*-control (*M*=4) shift register

four registers for three digits of information and four (or four sets of) controls. To shift *N* digits of information, approximately $4N/3$ register circuits are needed, and every control needs to drive $N/3$ loads. If the shift rate is limited by the delay of the register circuits, the conventional one has twice the maximum data rate as the present one. If the delay of the control drivers is the limiting factor, the difference in data rate is then less than a factor of 2 because of fewer loads to be driven by each clock.

The generalized results of the example above are: With *M* controls (or *M* sets of controls, $M > 2$) an *N*-digit shift register needs approximately $NM/(M-1)$ register circuits ($N \gg M$), as compared with $2N$ circuits in the conventional ones. The extreme of this approach is that $N+1$ registers and controls are needed to shift *N* digits of information. A slightly different approach but with the same limit has been used in the design of MOS dynamic random-access memories.^{7,10} In terms of circuit count, the number of register circuits is reduced by $(M-2)/2(M-1) \times 100$ percent. The controls, on the other hand, increased from two to *M*. To obtain a net benefit from this approach, the hardware cost of the added controls has to be small compared with the saving on registers. This is justified in most cases if a large quantity of shift registers is used in a system.

In LSI technology the gain of circuit reduction by using more controls is again related to two factors: (1) the area reduction of shift register chips, and (2) the cost for more inputs/outputs on the chip for the added control lines. The area reduction of the chip is always smaller than the $(M-2)/2(M-1) \times 100$ percent (reduction of the circuit count). Two factors contribute to this: (1) The added controls always cost some chip area. (2) Shift registers occupy a major portion of a

chip but not the complete chip. Some of the area is used by control circuits and pads for inputs/outputs. Only the array area occupied by the shift registers will be considered in this paper.

The chip area needed to accommodate the added control lines is a function of the register circuit configurations, the ground rules of the chip layout, and the device technology. To obtain meaningful general results, the following assumptions are used as the model of analysis:

1. The direction of data flow on the chip is perpendicular to the physical lines of the control signals.
2. The area taken by a single register circuit is increased δ times if an additional control line passes over it.
3. No crossovers are permitted among control lines.

All these assumptions are more or less based on single-layer metal interconnections being used on the chip.

Figure 3 shows an example of the model. Controls *A* and *C* are commonly used by both the upper and low row of registers and thus no additional area is needed for these two lines. Control lines *B* and *D*, however, do cost some additional area. When *M* is even, a maximum of two controls do not take additional area. When *M* is odd, only one control does not cost extra area. The total array area reduction thus depends on whether *M* is even or odd:

$$\begin{aligned} \text{Percentage Area Reduction} &\simeq \frac{M-2-(M-1)\delta}{2(M-1)} \\ &\times 100 \text{ percent} \\ M &= \text{odd number} > 2 \end{aligned} \quad (1)$$

$$\begin{aligned} \text{Percentage Area Reduction} &\simeq \frac{(M-2)(1-\delta)}{2(M-1)} \\ &\times 100 \text{ percent} \\ M &= \text{even number} \geq 2 \end{aligned} \quad (2)$$

These two equations are plotted in Figure 4. At $\delta = 0$, the area reduction ranges from 25 percent at $M = 3$ to its asymptotic value 50 percent as *M* increases. The most significant point shown in this figure is that the reduction of array area is rather insensitive to δ , the fractional area increase of an individual register circuit. This is especially true when *M* is even. For example, by reorganizing a two-control into a four-control shift register, 20 percent array area reduction can be achieved even if two more controls cause 40 percent area increase for half of the register circuits.

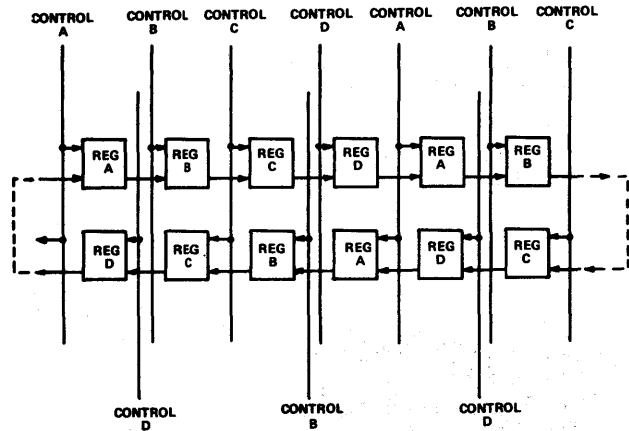


Figure 3—A possible topological arrangement of integrated shift registers where the information flow is perpendicular to the control lines

No distinction has been made so far on whether “*M* controls” stand for *M* clock signals or *M* sets of clock signals. Applications of the new approach on circuits given in Refs. 3-8 all support one result: the approach works best for register circuits that have only one control clock per gated register circuit. Among the circuits given in the references, the bucket-brigade shift register⁶ and circuits similar to the two-phase capacitor-pullup circuit⁴ obtain the most area improvement. The exact percentage of improvement depends on the ground rules of the device process. As an example, the bucket-brigade shift register is organized into a two-clock version and a four-clock version, as shown in Figures 5a and 5b. Conventional MOS technology is used.⁸ In the four-clock version, half the circuits have 30.8 percent area increase because of the two additional clocks, but the array area is reduced 23 percent with respect to the two-clock version.

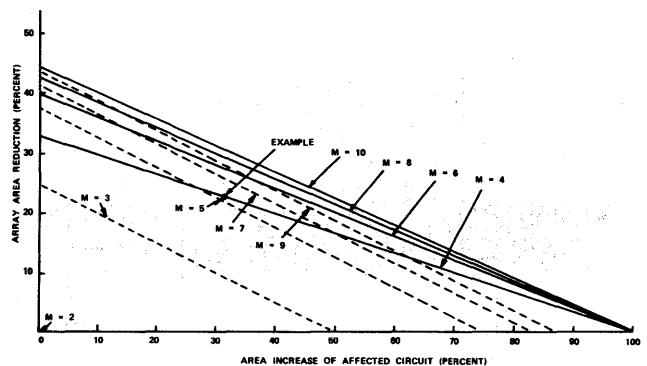


Figure 4—Array area reduction vs circuit area increase for different number of controls

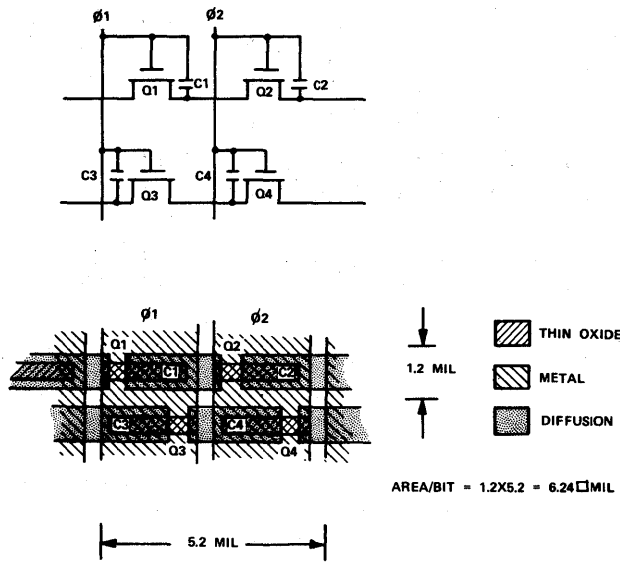


Figure 5a—Two-clock version of bucket-brigade shift register

PARALLEL OPERATION WITH SAMPLED INPUT/OUTPUT

As mentioned in the previous section, the maximum data rate is reduced as a result of the multicontrol approach. To overcome this shortcoming, parallel operation with sampled input/output can be used. Figure 6 illustrates a 4-control shift register system organized in such a way that the inputs and outputs are sampled by the controls of the shift register (and no additional gating signals are needed). The inputs of the two parallel shift registers are directly dotted together. This is allowed because the registers are gated by different controls. The outputs are also dotted directly together. This is allowed for most FET dynamic shift registers. Physically, these two shift registers can be

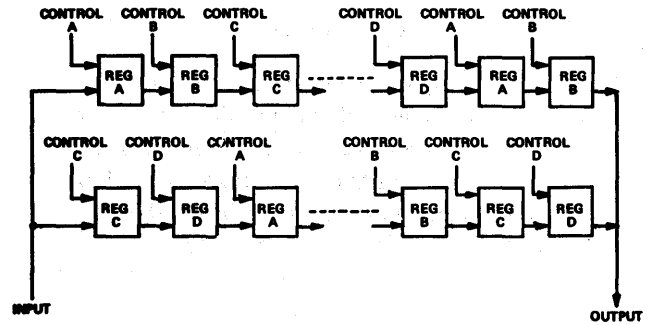


Figure 6—Parallel operation with sampled input/output

implemented side by side on the chip, or as two separate ones, either on the same chip or on different chips with no area penalty. The internal shifting speed is not changed by using this scheme, but the overall maximum data rate is doubled in this example. This, in effect, recovers the maximum data rate of conventional shift registers, provided the register speed determines the maximum data rate. In most practical designs, the drivers of the control clocks determine the maximum data rate.¹¹ Since the driver for *M*-control shift register drives considerably less load, higher maximum data rate is actually achieved by using this approach. In general *M*/2 *M*-control shift registers in parallel operation will have the same or higher data rate than that of the conventional one, provided *M* is an even number. There is no simple way to gain back the maximum data rate if *M* is an odd number.

The minimum data rate in FET dynamic shift registers is determined mostly by the leakage from the storage capacitors. A conventional shift register and its *M*-control version have the same minimum data rate if no parallel operation is used. For parallel operated *M*-control FET shift registers, the minimum data rate is approximately *M*/2 times higher. At a given data rate within the operating range, however, the parallel operated *M*-control shift registers consume approximately (*M* - 1) times less power than the conventional ones.

IMPLICATIONS ON DEVICE FABRICATION

Because of the higher minimum data rate, the present schemes require lower leakage if both the maximum and the minimum data rate are required. On the other hand, since the new schemes offer smaller chip area, a higher device process-yield is expected. Assuming that *Y*₀ is the yield of a conventional shift register array, then based on pure random defects, its *M*-control version

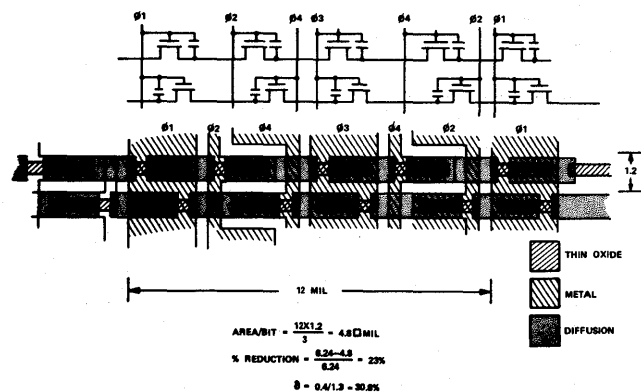


Figure 5b—Four-control version of bucket-brigade shift register

should give the following yield:¹²

$$Y \simeq (Y_0)^{[M+(M-2)\delta]/2(M-1)} \quad (3)$$

A very interesting situation exists when the M -control shift register is operated in parallel with sampled input/output. Assume that two 64-digit 4-control shift registers are operated in parallel to form a 128-digit shift register. One of the shift registers has a defect and is stuck to a fixed state. In the conventional shift register, one bad register ruins the whole shift register. In the present case, there is a good chance that a 64-digit shift register is still usable, assuming proper controls are used. For instance, by using discretionary wiring to connect the parallel operation, a good 64-bit shift register may have a good chance to be recovered. For $M=4$, the upper bound of the probability of recovering one shift register of half length is approximately

$$Y = 2 \times Y_0^{1/6(2+\delta)} (1 - Y_0^{1/6(2+\delta)}) \quad (4)$$

Figure 7 shows the plot of both Eqs. (3) and (4) at $M=4$ for different δ values as a parameter. Based on this model, better than 50 percent yield improvement can be achieved for a 20 percent original yield. In addition, there is up to 50 percent probability that a bad

shift register will have a recoverable good half-length shift register. For a more realistic yield model, the absolute value may be lower, but the general trend is not expected to change much.

CONCLUSIONS

This paper has described a new approach to improve the integrated shift registers by using different logic organization. By increasing the number of controls from 2 (conventional ones) to M , new shift registers can be organized which potentially offer many advantages. Higher density, higher yield, fewer number of circuits, and lower power consumption are among the most important potential improvements over using conventional shift registers. The study also shows that the high recovery of partial length M -control shift registers can be expected as a byproduct of this approach.

In practical applications of the new approaches, trade-offs are to be considered. The gain in density, in yield, and possibly in data rate have to be weighed against the increased number of controls, the number of I/O interconnections, and the device implications. The basic idea presented in this paper, however, is quite general. The reduction of circuit count is always true and is independent of the type of shift register. It is therefore expected that this work can be applied to designs of all shift registers.

The author thanks N. G. Vogl, Jr. for motivating this study, W. D. Pricer and I. T. Ho for many valuable discussions, and R. A. Henle and W. Hoffman.

REFERENCES

- 1 R W BOWER H G DILL K G AUBUCHON
S A THOMPSON
MOS field-effect transistors formed by gate masked ion implantations
IEEE Transactions on Electronic Devices Vol ED-15
No 10 1968
- 2 L L VADASZ A S GOVE T A ROWE
G E MOORE
Silicon gate technology
IEEE Spectrum Vol 6 No 10 1969
- 3 R L PETRITZ
Current status of large-scale integration technology
IEEE J Solid State Circuits Vol SC-2 No 4 1967
- 4 B G WATKINS
A low-power multiphase circuit technique
IEEE J Solid State Circuits Vol SC-2 No 4 1967
- 5 Fairchild Semiconductor 3320 Product Description
January 1968
- 6 F L J SANGTER
Integrated MOS and bipolar analog delay lines using bucket-brigade capacitor storage
Digest of Technical Paper 1970 ISSCC February 1970

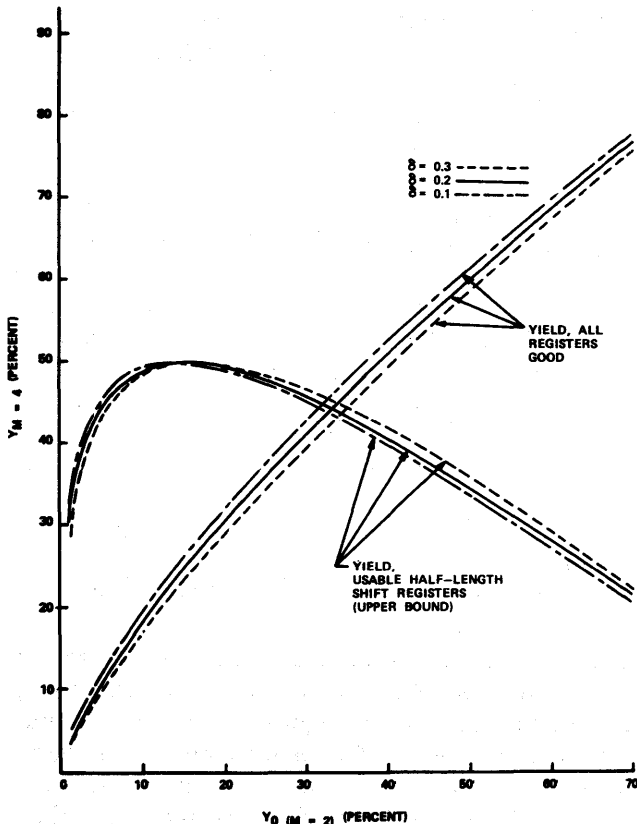


Figure 7—Yields of full and half shift registers for $M=4$

7 L BOYSEL W CHAN J FAITH

Random-access MOS memory

Electronics Vol 43 No 4 1970

8 F FAGGIN T KLEIN

A faster generation of MOS devices with low thresholds

Electronics Vol 42 No 20 1969

9 G A MALEY M F HEILWEIL

Introduction to digital computers

Prentice-Hall 1968

10 W M REGITZ J KARP

A three-transistor-cell 1024 500 ns MOS RAM

1970 ISSCC Digest of Technical Papers Vol 8

11 M E HOFF S MAZOR

Operation and application of MOS shift register

Computer Design Vol 10 No 2 1971

12 E TAMMARU J B ANGELL

Redundancy for LSI yield enhancement

IEEE J Solid-State Circuits Vol SC-2 No 4 1967

Universal logic modules implemented using LSI memory techniques

by KENNETH JAMES THURBER and ROBERT ORVAL BERG

*Honeywell Systems and Research Center
St. Paul, Minnesota*

INTRODUCTION

Large arrays of read only storage (ROS, ROM) are currently available from semiconductor vendors;^{1-4,10,11,14,15} however, there is a lack of material describing potential uses of these blocks of memory. This is due in part to the haste with which the semiconductor manufacturers place these devices on the market and in part to the fact that engineers have not yet realized the full potential presented by these devices for use as logic.

Previous researchers^{5,6,17-19} have considered how to use ROMs as logic devices; however, some of these approaches have not been fully pursued.^{5,16,17} Other approaches do not take full advantage of the capabilities offered by the application of such devices.^{6,16,17}

In Reference 7 Graham points out that one possible partial solution to the large costs and long development times associated with custom LSI chips is to replace random logic designs with ROM; however, this is probably only part of the total solution. Texas Instruments²⁰ has developed a Programmable Logic Array (PLA) which could possibly provide one economical solution to the random logic problem. Semiconductor manufacturers have described properties of ROM and some simple applications, but have never attempted to really investigate the power that is available through the use of ROM. Several variations of the ROM array have been introduced, notably, the ROAM and Solid Logic Technology (SLT) array.⁸ The SLT array may be modelled as a special case of the ROAM in which a variable can appear in either its complemented or uncomplemented form (but not both) and it appears in the same form in every minterm. Currently, 4096 bit MOS single chip ROMs are available.^{1,10,14,15} A single chip 8192 bit ROM is currently available.¹¹ In the next one to two years, it is conceivable that ROM with two to three times as many bits will be available. In fact, it is anticipated that every year for the next

five years, chip complexity will probably double⁷. By 1973, 16,384-bit single chip ROMs should be readily available. Prices are also anticipated to reflect this improved capability, and will probably be less than a penny a bit.⁴ This should enable the clever logic designer to economically perform extensive logic functions with memory! One obvious use of such devices is table lookup arithmetic; however, this is only one of the many ways they can be used. Four possible uses of read-only devices are given in this paper. These four uses were selected because they illustrate the power of read-only devices, show that "pie-in-the-sky" concepts such as Universal Logic Modules (ULM) may actually be possible, illustrate the future potential of read only devices, illustrate the failings and tradeoffs involved with several read only devices, and propose relatively simple solutions to several problems.

In the next section, use of read only devices in the construction of a universal combinational logic module is given. Next, use of read only devices in the construction of a universal sequential logic module is discussed. Then the use of read only devices in the realization of arbitrary sequential machines is presented. In all cases several different implementations of the devices under consideration are given. Finally, a new type of read only array is introduced which is a hybrid combination of ROM and ROAM. It is shown that for a multiple output function this hybrid implementation is most economical.

A PROGRAMMABLE UNIVERSAL COMBINATIONAL LOGIC MODULE

Figure 1(a) shows a section of ROM.* By choosing $V_E=1$ and $G=0$, it can readily be seen that if line *A*

* The ROM and ROAMs have been implemented using diodes in the examples presented in this paper; however, they could have been just as easily implemented using transistors.

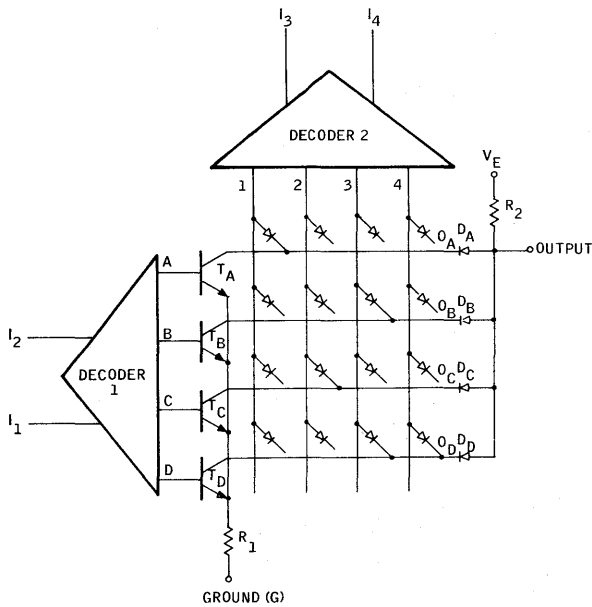


Figure 1(a)—A 16 bit read only memory

and line 1 are both 1 then the output line O_A is 1 if and only if the diode at the intersection of lines A and 1 is connected to both lines. (A more detailed explanation of the working of ROM and ROAM can be found in the appendix.) This then allows a designer to specify

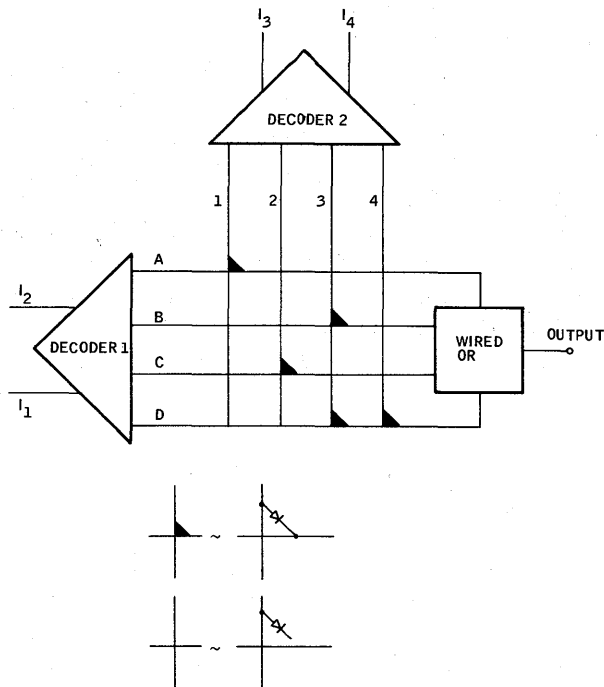


Figure 1(b)—Simplified representation of the ROM shown in Figure 1(a)

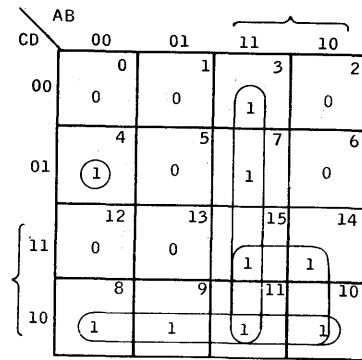


Figure 2(a)—Karnaugh map of $AB + CD + AC + \bar{A}\bar{B}\bar{C}D$

the function that is programmed into an array. One practical method of utilizing this is in generating a logic function. In order to ensure that exactly one vertical line and one horizontal line are 1 at the same time, decoders are used. The circuit of Figure 2 shows two ways in which a four-variable function can be implemented. If the input combination $A^*B^*C^*D^*$ is to yield a 1 then the diode is connected to the appropriate horizontal and vertical lines. The circuit shown in Figure 2 performs the logic function $f = AB + CD + AC + \bar{A}\bar{B}\bar{C}D$. Several articles have pointed out the relationship between the ROM (ROS) and the Karnaugh Map.^{6,8,17}

Figure 3 illustrates two versions of the read only associative memory (ROAM).*** In Figure 3(a), if

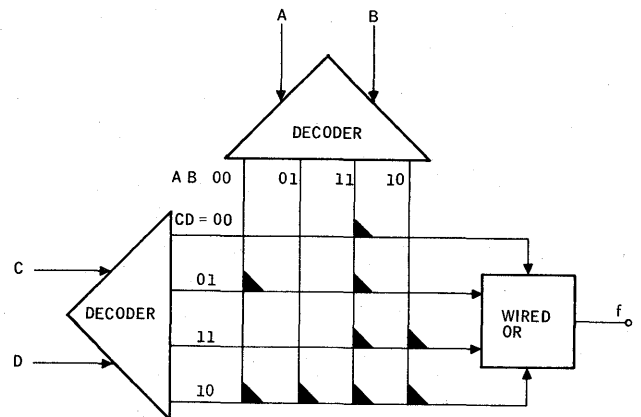


Figure 2(b)—Implementation of $AB + CD + AC + \bar{A}\bar{B}\bar{C}D$ in a 16-bit ROM

** A^* denotes A or \bar{A} but not both. A^*B^* denotes exactly one of $AB, \bar{A}\bar{B}, \bar{A}B, A\bar{B}$.

*** R. A. Henle, et al., refer to the array in Figure 3(b) as the SLT array (Solid Logic Technology Array); however, this array can be modelled as a simplified version of the ROAM.

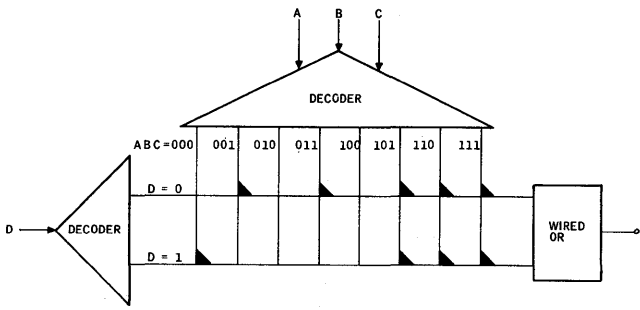


Figure 2(c)—An alternate implementation of $AB+CD+AC+ABCD$ in a 16-bit ROM

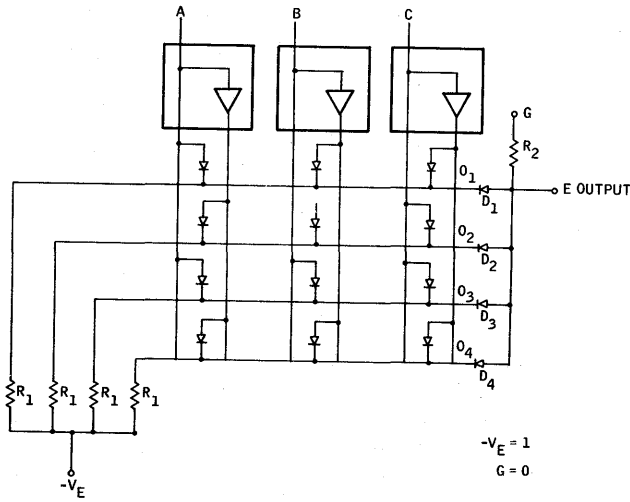


Figure 3(a)—Three input, one output ROAM1

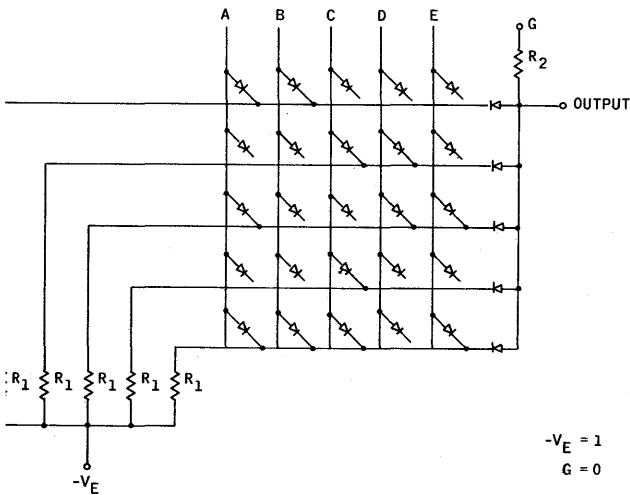


Figure 3(b)—Five input, one output ROAM2

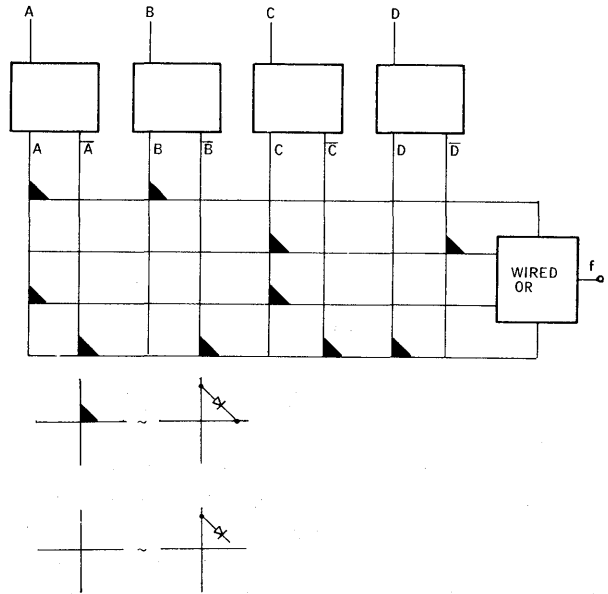


Figure 4(a)—ROAM1 realization of $AB+CD+AC+\bar{A}\bar{B}\bar{C}$

— $V_E=1$ and $G=0$ then it can be seen that the output line E is 1 if and only if the input word $A, B,$ and C exactly matches a word stored in the ROAM. This then allows the construction of logical product terms, one in each row of the ROAM. These product terms can then be wire ORed together to form a Boolean function in the sum of products form. Figure 4(a) illustrates the implementation of the function $F=AB+CD+AC+\bar{A}\bar{B}\bar{C}$ in ROAM1. A function such as $f=AB+A\bar{D}+D+C$ cannot be implemented in ROAM2 unless both of the variable D and \bar{D} are available on a separate line as shown in Figure 4(b). Therefore, ROAM2 is not quite as flexible as ROAM1, but it can be used for functions that do not require double rail logic for all variables.

Since ROM is completely programmable, any

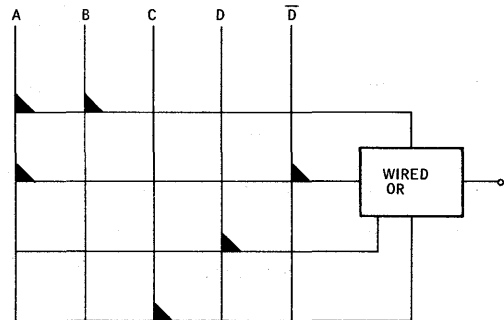


Figure 4(b)—ROAM2 realization of $AB+A\bar{D}+D+C$

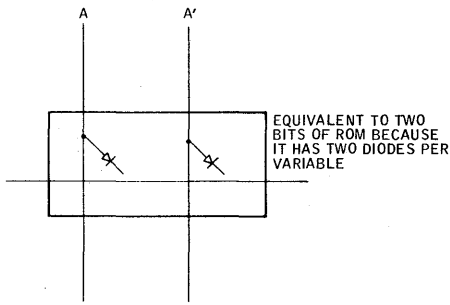


Figure 4(c)—A worst case ROAM1 cell

arbitrary logic function can be generated. The designer gives the manufacturer the outputs he desires for each allowable input combination. The manufacturer then programs the ROM by appropriately connecting the diodes (transistors) to the desired leads and the array is tested. Since the ROM can be programmed arbitrarily for its input conditions, the complexity of the logic function is not of great significance. Since the ROM can be used to generate arbitrary logic functions, it can be used to replace random logic in the control sections of computers. It then begins to seem feasible that computers can be constructed from only two elements; e.g., memory and a means for moving or transferring data. Memory will consist of ROM (or ROAM) performing all the logic and control functions and some

FUNCTION NUMBER	FUNCTION	LOGIC REPRESENTATION
0	QUAD TWO-INPUT AND	
1	DUAL FOUR-INPUT AND	
2	EIGHT-INPUT AND	
3	TWO DUAL TWO-INPUT AND, OR	
4	DUAL FOUR-INPUT AND, OR	
5	AND, OR INPUT RS FLIP FLOP	
6	TWO TWO-INPUT AND RS FLIP FLOP	
7	TWO TWO-INPUT AND JK FLIP FLOP	

Figure 5—The functions included in the combinational module

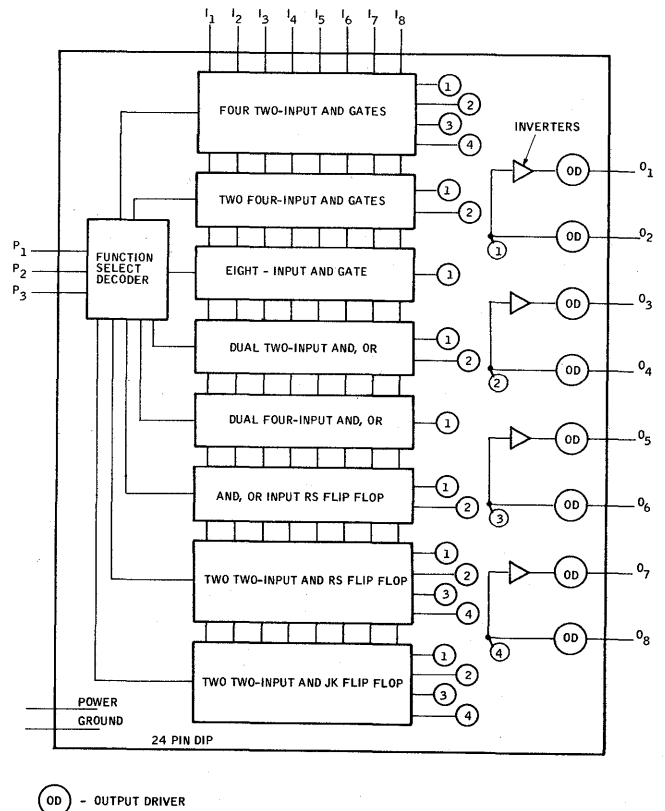


Figure 6—Block diagram of the module

type of read/write memory for storage. (Henle et al.,⁸ illustrate the use of ROAM and SLT arrays in generating logic functions and show some size comparisons between the three types of read only devices. A worst case realization of ROAM1 would require the equivalent of approximately 2 bits of ROM to form 1 bit of ROAM as shown in Figure 4(c).)

Function Performed	$P_1 P_2 P_3$	Inputs	Output Selection	Number
Quad 2-input AND	000	Normal	$O_1 O_2 O_3 O_4$	1
Quad 2-input NAND	000	Normal	$O_1 O_2 O_3 O_4$	2
Quad 2-input OR	000	Complemented	$O_1 O_2 O_3 O_4$	3
Quad 2-input NOR	000	Complemented	$O_1 O_2 O_3 O_4$	4
Quad Inverters	000	$I_1=I_2=I_3=I_4=1$ Rest Normal	$O_1 O_2 O_3 O_4$	5
Dual 4-input AND	001	Normal	O_2	6
Dual 4-input NAND	001	Normal	O_3	7
Dual 4-input OR	001	Complemented	O_3	8
Dual 4-input NOR	001	Complemented	O_2	9
8 input NAND	010	Normal	O_1	10
8 input AND	010	Normal	O_2	11
8 input OR	010	Complemented	O_1	12
8 input NOR	010	Complemented	O_2	13
Two dual two input AND OR	011	Normal	$O_2 O_4$	14
Two dual two input AND OR invert	011	Normal	$O_1 O_3$	15
Two dual two input NOR OR	011	Complemented	O_2	16
Two dual two input OR AND	011	Complemented	O_1	17
Dual 4 input AND OR	100	Normal	O_2	18
Dual 4 input AND OR invert	100	Normal	O_1	19
Dual 4 input NOR OR	100	Complemented	O_2	20
Dual 4 input OR AND	100	Complemented	O_1	21
AND OR Input RS flip-flop (complementary outputs)	101	Normal	$O_1 O_2 (O_2 O_1)$	22
Two input AND RS flip-flop (complementary outputs)	110	Normal	$O_1 O_2 O_3 O_4 (O_2 O_1 O_4 O_3)$	23
Two input AND JK flip-flop (complementary outputs)	111	Normal	$O_1 O_2 O_3 O_4 (O_2 O_1 O_4 O_3)$	24
Two trigger flip-flops (complementary outputs)	111	$I_1=I_2=I_3=I_4=1, I_5=I_6=I_7=I_8$	$O_1 O_2 O_3 O_4$	25

Figure 7—The twenty-five functions produced by the combinational module

Since MOS ROMs of 4096 bits are available^{1,10,14,15} and not much in the line of MOS random logic is available,[†] a MOS universal logic module has been designed using ROM and both types of ROAM.[‡] The functions that were selected for inclusion in the module are available from a number of manufacturers and are shown in Figure 5. The general design of the module is shown in Figure 6. One notes that both true and complementary outputs are available and therefore by selection of appropriate outputs and judicious use of complementary inputs, the module can produce 25 different major functions as shown in Figure 7.

The eight logic functions included in the package were chosen because these functions can allow the package to produce any one of 25 functions by properly inverting the input and output terminals. The means to invert the outputs has been provided internal to the package; however, it is assumed that the inputs would be provided in their proper format (inverted or normal) to the input of the package. In Figure 7 the functions performed by the same function generator but using

[†] In January 1970, Electronic Arrays, Inc. announced a line of MOS logic devices EA1800, EA1801, EA1802, EA1804, EA1806, and EA1808. These arrays are for use in generating random logic and are probably the most complete MOS line available at this time.

[‡] In realizing logic with ROM it is possible to produce economical realizations using more than one level of logic and/or by careful consideration of the function being produced. For example, function 0 (Zero) in Figure 5 is 4 (four) AND gates and could be realized as four 4 bit ROMs for a total of 16 bits instead of the 1024 bits required by the authors. In fact the whole module can be realized in ROM with only 304 bits needed to realize the functions. The equivalent amount of bits needed for decoding is near 1000 (same order of magnitude as in the solution given in Figures 8d and 9) and therefore this realization is still more costly than either ROAM solution. Additionally this solution would have other drawbacks. Some of these are the following:

- a large number of small decoders have been substituted for a large decoder thus increasing the amount of random wiring
- multiple level logic realizations yield multiple level "gate" delays thus making the speed of the functions in the module dependent upon the realization while all combinational functions (functions 0-4) in the module of Figure 8d (flip-flops are the exceptions) operate at the same speed (functions 5 and 6, the R-S flip flops, operate at the same speed as each other)

The fact that economical ROM realizations can be achieved using multiple level logic does not change the conclusions of this paper. Care must be taken in the application of ROM to insure that the most economical realization is used *consistent with other design goals and constraints*. In fact it is conceivable that the search for economical multi-level realizations could result in higher overall chip cost due to the increase in random wiring. Regularity of structure has been demonstrated to be a very important factor (if not the most important) in LSI by the very fact that memories and shift registers can be built in the large sizes that they are today, whereas, random logic LSI is limited to hundreds of gates.

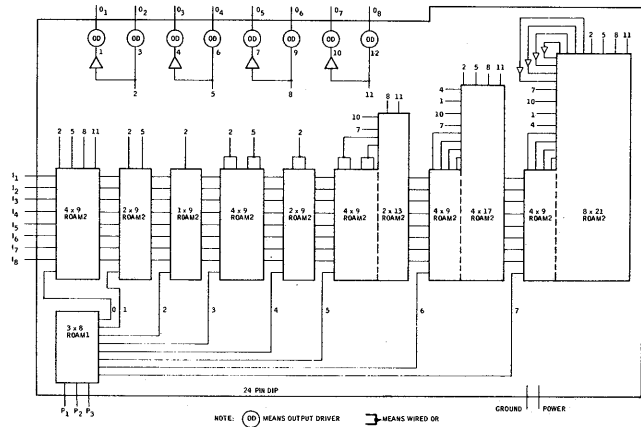


Figure 8(a)—Block diagram of the ROAM2 solution

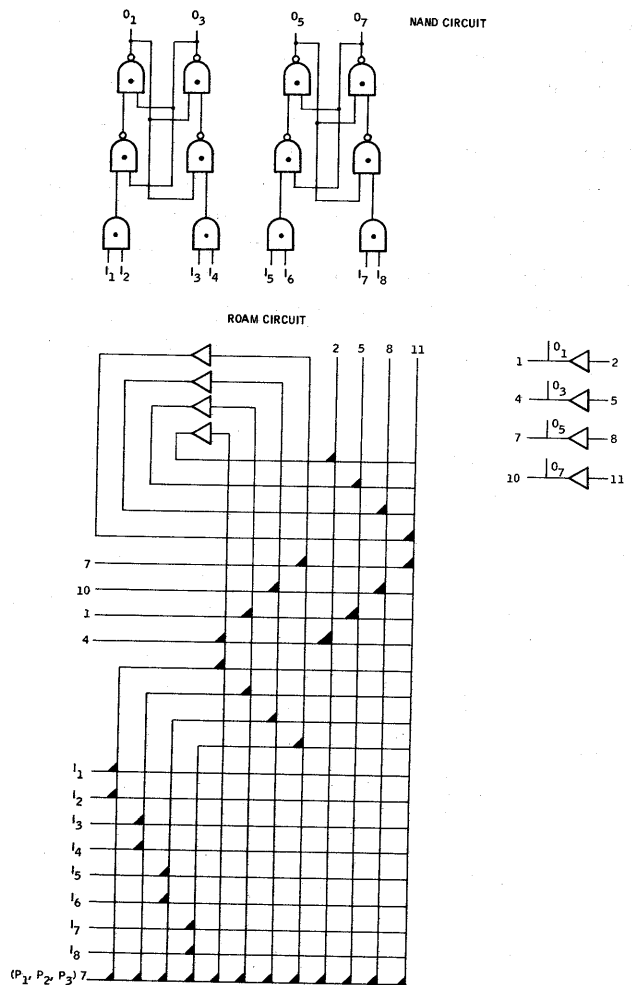


Figure 8(b)—Sketch of the ROAM2 implementation of two JK flip flops

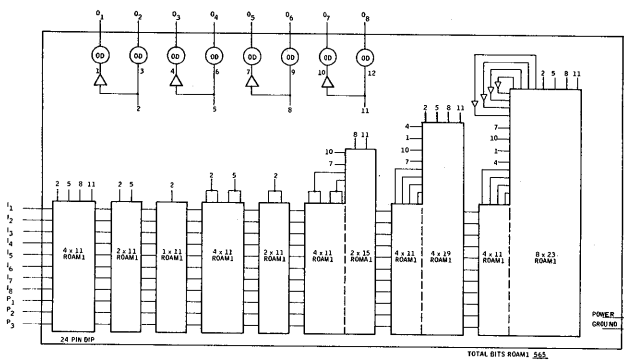


Figure 8(c)—Block diagram of the ROAM1 solution

inverted inputs and outputs have been bracketed for easy identification. It is felt that this one package could replace a large percentage (over 70 percent) of the random logic packages used today. Obviously, with enough of these packages, one could generate any arbitrary logic function.

While it is not expected that this package would ever be cheaper than standard present day packages which perform a *single logic function*, its final cost to the user should be less especially in small to medium quantity ranges. This results because you only have to purchase one package type, stock one package type, and have spares for one package type thus cutting inventory and handling costs. Rework costs on manufactured boards and assembly costs should be minimized because there

is only one type of package to be handled. Thus, although this module is slightly more costly than many present day single function ICs, its versatility should be considered in any comparisons. As total cost of ownership considerations become more prevalent, this concept should gain wide acceptance.

The implementation of these functions has been outlined in block diagram form for each of the two types of ROAM (see Figure 3(a) and 3(b)) and for ROM (see Figure 1) in Figure 8. Figure 8(b) shows how one part of one of the circuits (Function 7) could be implemented in ROAM. From the implementations it can be seen that the ROM solution is probably not able to be implemented today; however, it will be implementable in the near future. The ROAM solutions are implementable at this time. Figure 9 shows the number of bits needed to realize each function, based upon the designs in Figure 8. Since these devices would be produced in large quantity and are well suited for LSI production, the cost per bit projections for ROM shown in Figure 10 are applicable to this device.

Looking at the realizations presented in Figure 8, it can be seen that the manner in which the functions are realized and the nature of the functions (essentially AND gates) produces the cheapest realization from ROAM2, the next cheapest from ROAM1, and the most expensive realization from ROM. With current technology it should be possible to build the solutions of ROAM1 and ROAM2 in either MOS or bipolar technology. The solution for ROM is presently pushing the state of the art; however, if the dual RS flip flop

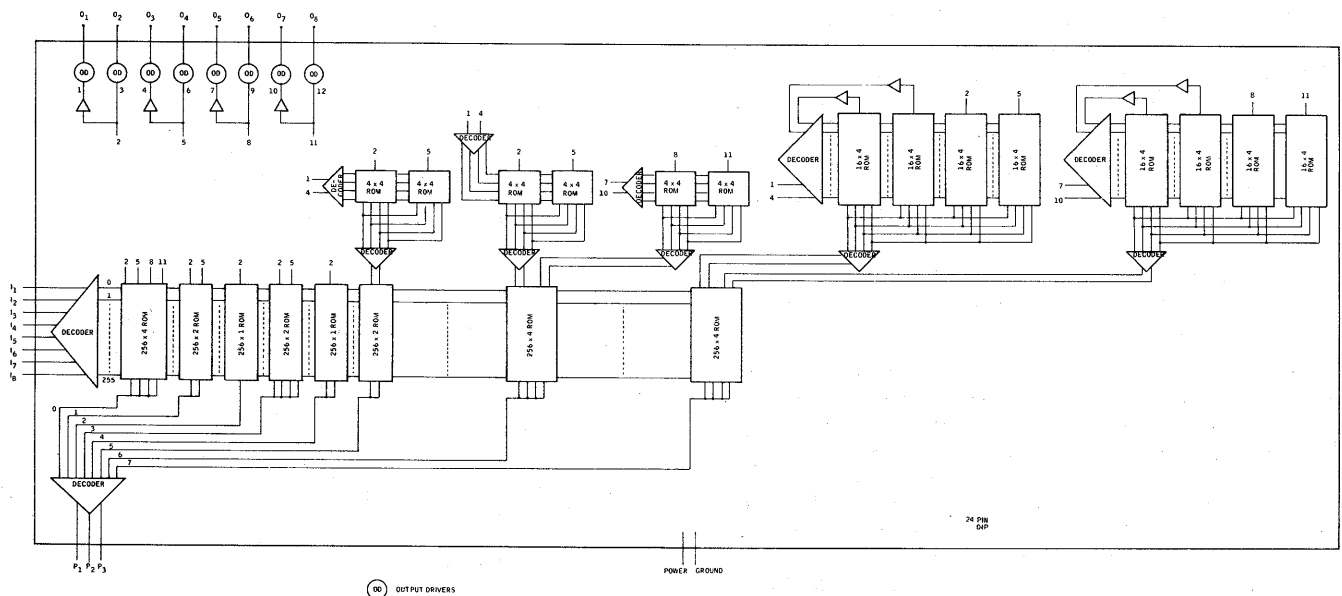


Figure 8(d)—Block diagram of the ROM solution

Note: 2 ROM bits are equivalent to 1 ROAM1 bit. All bit totals are given in terms of bits of ROM.

ROAM2		ROAM1	
Function	# of Bits	Function	# of Bits
0	36	0	64
1	18	1	32
2	9	2	16
3	36	3	64
4	18	4	32
5	62	5	112
6	104	6	192
7	204	7	384
Decoder	48	Decoding	234
Total Bits	535	Total Bits	1130

ROM	
Function	# of Bits
0	1024
1	512
2	256
3	512
4	256
5	544
6	1088
7	1536
Decoder	1336
Total Bits	7064

Figure 9—Number of bits required to realize the combinational module

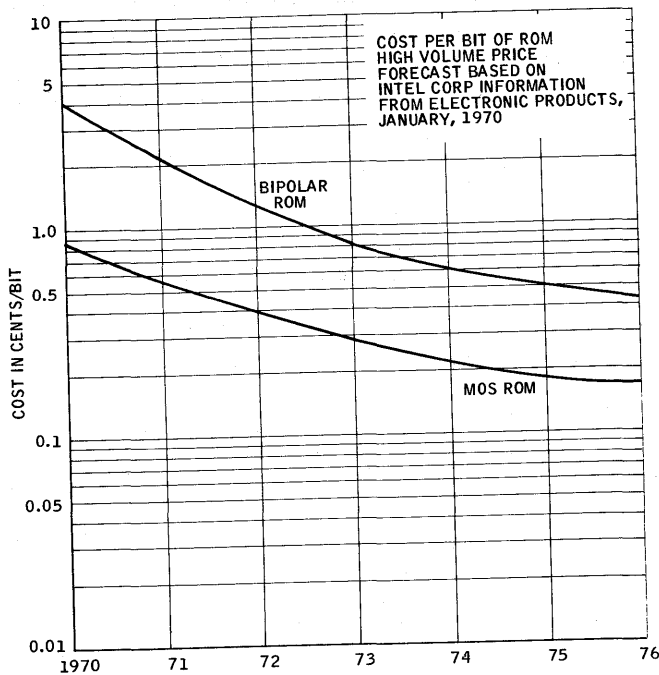


Figure 10—Cost per bit of ROM

(or the dual JK flip flop) were left out, this can probably be built. All of the solutions can be packaged in a 24-pin DIP. One problem with this implementation is that the outputs must not change while the new information is being read. This can be assured in MOS implementations because of the capacitance; however, in a bipolar implementation it may be necessary to add a clock to the system and include a clocked latch flip flop in each output driver.

A PROGRAMMABLE SEQUENTIAL LOGIC MODULE

A programmable sequential logic module is defined and implemented in this section. This module is programmable and can perform three different functions. This module is based upon widely available IC logic series and it is felt that this module can perform the most important sequential functions. This module is designed so that it can be used with itself and the ULM from the previous section in the implementation of more complex systems and subsystems.

Figure 11 is a flow table description of the functions that can be performed by the module. The three functions implemented in the module are three different

F D B K	DC			
	0 0	0 1	1 0	1 1
0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 1	1 1 1 1
0 0 0 1	0 0 0 1	0 0 0 0	0 0 1 0	0 0 0 0
0 0 1 0	0 0 1 0	0 0 0 0	0 0 1 1	0 0 0 1
0 0 1 1	0 0 1 1	0 0 0 0	0 1 0 0	0 0 1 0
0 1 0 0	0 1 0 0	0 0 0 0	0 1 0 1	0 0 1 1
0 1 0 1	0 1 0 1	0 0 0 0	0 1 1 0	0 1 0 0
0 1 1 0	0 1 1 0	0 0 0 0	0 1 1 1	0 1 0 1
0 1 1 1	0 1 1 1	0 0 0 0	1 0 0 0	0 1 1 0
1 0 0 0	1 0 0 0	0 0 0 0	1 0 0 1	0 1 1 1
1 0 0 1	1 0 0 1	0 0 0 0	1 0 1 0	1 0 0 0
1 0 1 0	1 0 1 0	0 0 0 0	1 0 1 1	1 0 0 1
1 0 1 1	1 0 1 1	0 0 0 0	1 1 0 0	1 0 1 0
1 1 0 0	1 1 0 0	0 0 0 0	1 1 0 1	1 0 1 1
1 1 0 1	1 1 0 1	0 0 0 0	1 1 1 0	1 1 0 0
1 1 1 0	1 1 1 0	0 0 0 0	1 1 1 1	1 1 0 1
1 1 1 1	1 1 1 1	0 0 0 0	0 0 0 0	1 1 1 0

DC = 00 stop
 DC = 01 reset
 DC = 10 count up
 DC = 11 count down

Figure 11(a)—Flow table for a binary counter

I ₀ I ₁ I ₂ I ₃	PDC			
	1 X X	0 0 X	0 1 1	0 1 0
0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 1	1 0 0 1(1)
0 0 0 1	0 0 0 1	0 0 0 1	0 0 1 0	0 0 0 0
0 0 1 0	0 0 1 0	0 0 1 0	0 0 1 1	0 0 0 1
0 0 1 1	0 0 1 1	0 0 1 1	0 1 0 0	0 0 1 0
0 1 0 0	0 1 0 0	0 1 0 0	0 1 0 1	0 0 1 1
0 1 0 1	0 1 0 1	0 1 0 1	0 1 1 0	0 1 0 0
0 1 1 0	0 1 1 0	0 1 1 0	0 1 1 1	0 1 0 1
0 1 1 1	0 1 1 1	0 1 1 1	1 0 0 0	0 1 1 0
1 0 0 0	1 0 0 0	1 0 0 0	1 0 0 1	0 1 1 1
1 0 0 1	1 0 0 1	1 0 0 1	0 0 0 0(1)	1 0 0 0
1 0 1 0	1 0 1 0	1 0 1 0	0 0 0 1	0 0 0 1
1 0 1 1	1 0 1 1	1 0 1 1	0 0 0 0	0 0 0 0
1 1 0 0	1 1 0 0	1 1 0 0	0 0 0 1	0 0 0 1
1 1 0 1	1 1 0 1	1 1 0 1	0 0 0 0	0 0 0 0
1 1 1 0	1 1 1 0	1 1 1 0	0 0 0 1	0 0 0 1
1 1 1 1	1 1 1 1	1 1 1 1	0 0 0 0	0 0 0 0

PDC = 1XX preset
 PDC = 00X hold
 PDC = 011 count up
 PDC = 010 count down

() value in fifth column indicates the output value when it becomes 1. This is the "carry out" D₀.

Figure 11(b)—Flow table for a decade counter

I ₀ I ₁ I ₂ I ₃	PDC			
	1 X X	0 0 X	0 1 1	0 1 0
0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 1	1 1 1 1(1)
0 0 0 1	0 0 0 1	0 0 0 1	0 0 1 0	0 0 0 0
0 0 1 0	0 0 1 0	0 0 1 0	0 0 1 1	0 0 0 1
0 0 1 1	0 0 1 1	0 0 1 1	0 1 0 0	0 0 1 0
0 1 0 0	0 1 0 0	0 1 0 0	0 1 0 1	0 0 1 1
0 1 0 1	0 1 0 1	0 1 0 1	0 1 1 0	0 1 0 0
0 1 1 0	0 1 1 0	0 1 1 0	0 1 1 1	0 1 0 1
0 1 1 1	0 1 1 1	0 1 1 1	1 0 0 0	0 1 1 0
1 0 0 0	1 0 0 0	1 0 0 0	1 0 0 1	0 1 1 1
1 0 0 1	1 0 0 1	1 0 0 1	0 0 0 0(1)	1 0 0 0
1 0 1 0	1 0 1 0	1 0 1 0	0 0 0 1	0 0 0 1
1 0 1 1	1 0 1 1	1 0 1 1	0 0 0 0	0 0 0 0
1 1 0 0	1 1 0 0	1 1 0 0	0 0 0 1	0 0 0 1
1 1 0 1	1 1 0 1	1 1 0 1	0 0 0 0	0 0 0 0
1 1 1 0	1 1 1 0	1 1 1 0	0 0 0 1	0 0 0 1
1 1 1 1	1 1 1 1	1 1 1 1	0 0 0 0(1)	1 1 1 0

PDC = 1XX preset
 PDC = 00X hold
 PDC = 011 count up
 PDC = 010 count down

() value in fifth column indicates the output value when it becomes 1. This is the "carry out"

Figure 11(c)—Flow table for a presetable binary counter

counters. A four-bit binary counter (Figure 11(a)) has been implemented which has the capability to count up or down with a range of 16 different numbers; i.e., 0 to 15. This counter can also be reset to zero.

The second counter (Figure 11(b)) is an up-down BCD decade counter. This counter can count up or down and can also be preset to a value. This counter is constrained to count between zero and nine and can be ganged with other decade counters to form a counter capable of counting between zero and 999 . . . 9.

The last counter (Figure 11(c)) is an up-down, four-bit binary counter. This counter can be preset to a value and is constrained to count between zero and 15. This counter can also be ganged with other binary counters. The differences between the first counter and this counter are that this counter can be preset to any value and ganged with other binary counters.

The module is contained in a 24-pin DIP. Figure 12 is a block diagram of the overall layout of the package. This diagram shows the necessary leads for the package and the general functions that the package performs. The module was realized in ROM and ROAM1. No attempt was made to realize the function in ROAM2 since double-rail logic was necessary. The realizations are given in Figure 13. Since the functions are realized

in a sum-of-products form, it would be natural to simplify the functions as if the function to be produced were a multiple-output function; however, this is impossible since sharing of terms between functions does not work properly when the wired-OR is used.

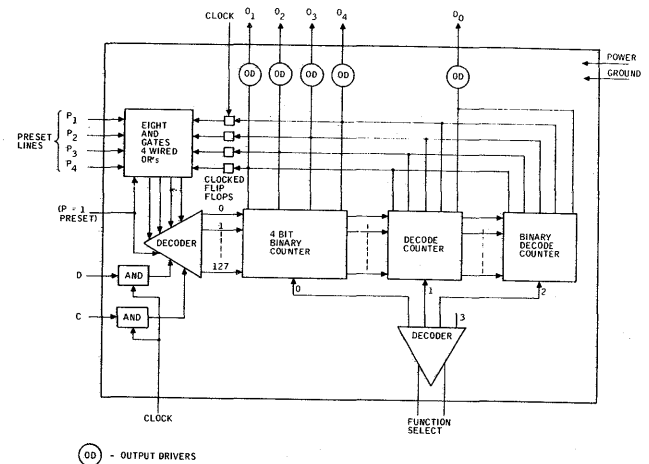


Figure 12—Block diagram of the sequential module

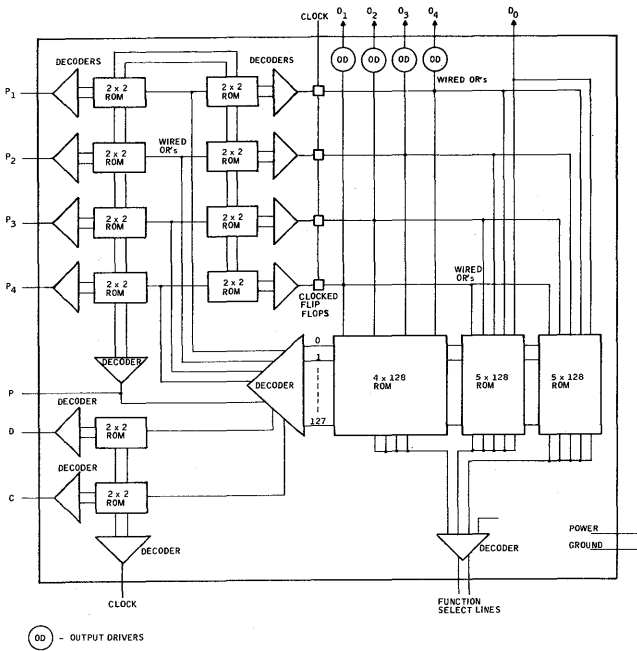


Figure 13(a)—ROM solution for the sequential module

Therefore, the functions were realized as single-output functions.

The number of equivalent bits of ROM used by the solutions given in Figure 13 is shown in Figure 14. The parenthesized number in the ROAM1 solution is an alternative bit count based upon the use of a different method of decoding. Instead of selecting the function at each implicant requiring $2 \times 2 \times 74 = 296$ bits of ROM

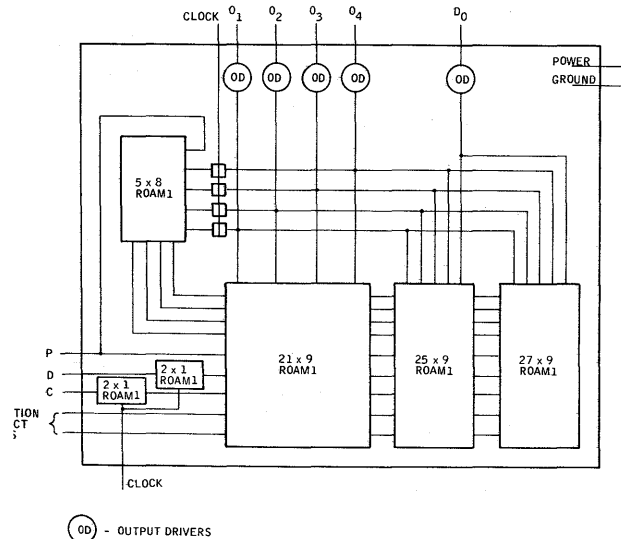


Figure 13(b)—ROAM1 solution for the sequential module

ROM		ROAM1	
Function	Cost	Function	Cost
1	512 (256)	1	294
2	640	2	350
3	640	3	378
Decoders	320	Decoders	296 (160)
Misc.	40	Misc.	80
	<u>2152 (1896)</u>		<u>1406 (1270)</u>

Figure 14(a)—Number of bits required for the sequential module

as shown in Figure 13(b) the 160 is based upon the construction of a decoder requiring three implicant terms at a cost of $3 \times 2 \times 2 = 12$ plus the cost of one control bit per implicant $2 \times 1 \times 74 = 148$ or a total of 160 bits. The control line from each implicant term would then pass through only the portion of ROAM1 it selects just as the ROM solution decoder lines do in Figure 13(a).

In the ROM solution, 256 bits of ROM can be saved because the first function does not depend on the

Implicants necessary for function 1

Variable	# Implicants
F	8
D	6
B	5
K	<u>2</u>
	21

Implicants necessary for function 2

Variable	# Implicants
D ₀	2
I ₀	6
I ₁	7
I ₂	7
I ₃	<u>3</u>
	25

Implicants necessary for function 3

Variable	# Implicants
D ₀	2
I ₀	9
I ₁	7
I ₂	6
I ₃	<u>3</u>
	27

Figure 14(b)—Number of implicants required for the counters

variable P . It is necessary that $P=0$ in order for the first function to operate correctly; therefore, it is conceivable to build the first function using only 256 bits of ROM instead of 512 bits since only 256 of the 512 bits are used anyway.

Cost projections for this device can be prepared using Figure 10. No timing problems are anticipated with this universal sequential logic module. In order to alleviate the chance of any races occurring, clocked flip flops have been provided as delay units; however, even in the event the module was built without providing a means for the flip flops to be clocked, no problems are anticipated. It is felt that ample time can be allowed so that the network could obtain a stable state without jeopardizing the module's speed. In order to obtain this objective, the feedback variables and their delay hardware were kept completely internal to the module. The speed of each feedback signal is approximately equal. It is anticipated that if the flip flop delay units were placed outside of the module and no feedback occurred inside the module that serious timing problems could occur because of the input-output process to the package and the inherent characteristic differences in the various parts and this could reduce the module's speed by over 50 percent.

It should be noted that many different functions could be placed on the chip. One might want to consider counters such as a divide-by-six counter or a divide-by-four counter; however, because of the level of complexity that can be reached in this type of logic, the authors see no real advantage in placing too many counters on the chip because they are so much alike. A good understanding of the ideas from which these chips have been designed should allow anyone to define his own complete ULM (Universal Logic Module) that can be easily realized.

USE OF READ-ONLY MEMORIES IN REALIZING ARBITRARY SEQUENTIAL MACHINES

In previous sections it has been shown that ROM and ROAM can be successfully applied to the design of both universal combinational logic modules and sequential logic modules that replace a large number of ICs. These modules are functionally programmable by setting different values on the control lines. Because of the logical power of ROM and ROAM, it seems that these devices may hold the key to the ability to realize arbitrary logic networks by means of structured or regular logic. Therefore, in this section it will be shown that it is possible to formulate design guidelines that

will enable the designer to describe a regular module that is capable of exhibiting an arbitrary sequential behavior pattern. Since this sequential machine can be built from ROM and ROAM, it is possible to build machines in a standard form and then program each machine to perform its specific function by specifying the interconnection of the ROM and ROAM. Bounds on the size of flow table that currently can be realized are also given.

It is felt that there are many practical advantages that can be achieved by using this method of realizing a sequential machine on a single chip. The four most important advantages are the following: (1) it is possible to minimize the number of input and output leads since the feedback (next state) variables can be handled internal to the chip, thereby yielding a high gate-to-pin ratio; (2) an iterative realization of the machine will minimize (and possibly eliminate) hazards and race conditions because there is very little time lag to be accounted for due to the extremely close proximity of all devices; (3) because there are no distinct parts to be connected together, there will be no large line delays

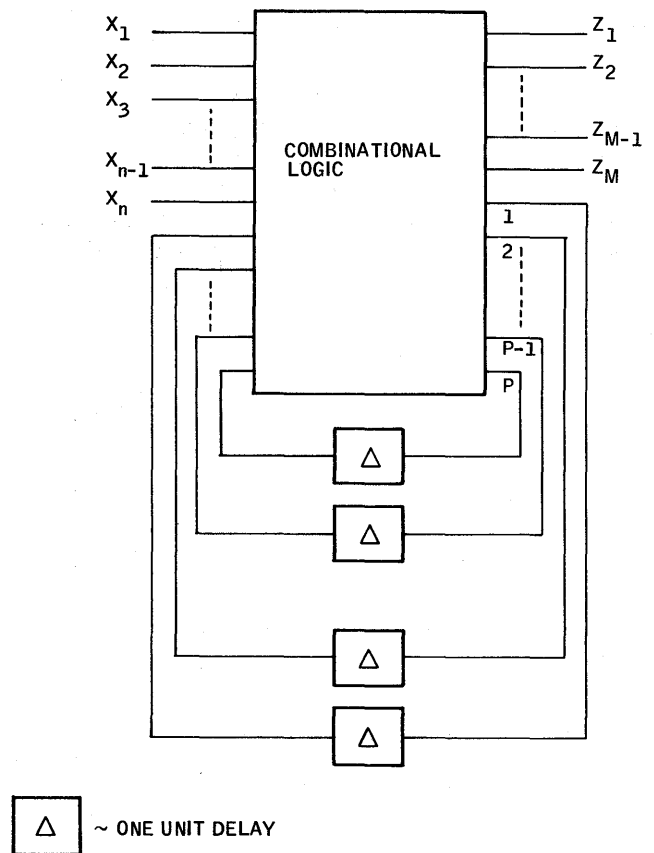


Figure 15—Block diagram of a sequential machine

thus yielding an extremely fast machine; and (4) it is conceivable to be able to realize complex special-purpose sequential machines out of mass-produced chips by just programming the chip in the near future. The concept of placing a subsystem on a chip is felt to be the logical extension of ULM. Therefore, a discussion of bounds on the size of the system is provided in this article.

Figure 15 shows a block diagram of the class of sequential machine under consideration. This machine has n inputs, m outputs, and p next-state functions. This gives the machine a capability of realizing any 2^p state machine with n inputs and m outputs where each of the output functions and next-state functions is a function of n external variable and p next-state variables.

Figures 16 and 17 show the manner in which this type of network could be organized on a chip. The flip flops (FF) and AND gates could be mechanized using ROM or ROAM and the decoder could be mechanized using ROAM. As seen from the figures, any mechanization must have $m+n+3$ package leads. This allows one lead for power, one for ground, one for the system clock, n input leads, and m output leads.

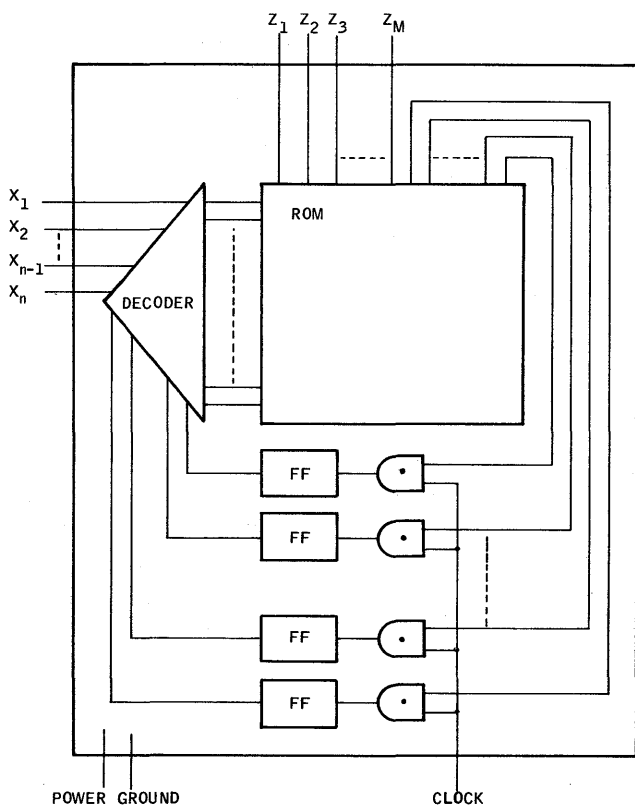


Figure 16—Block diagram of the ROM implementation of a sequential machine

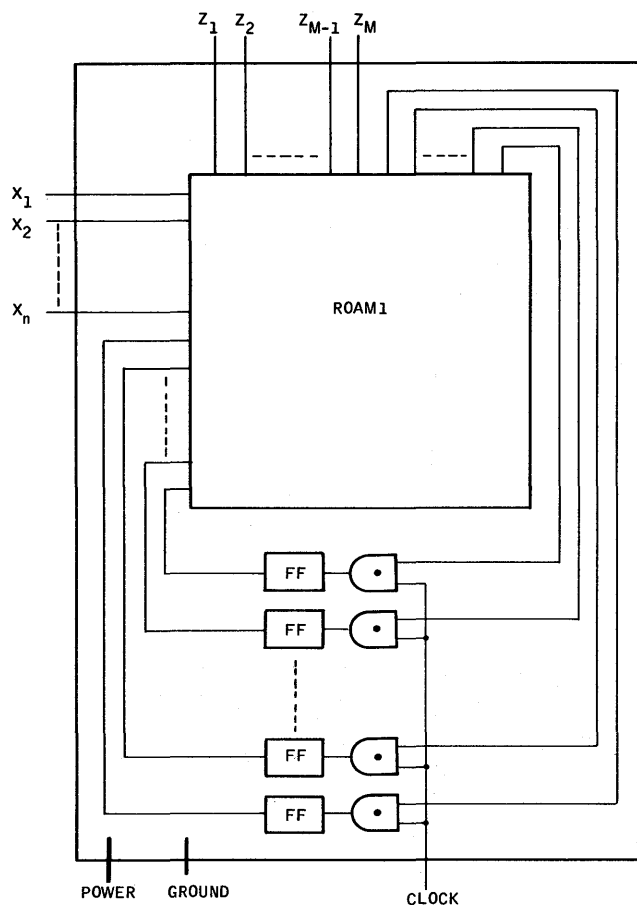


Figure 17—Block diagram of the ROAM1 implementation of a sequential machine

Internal to the package are p flip flops and the necessary gates to clock the next states into the flip flops. Since the machine has p next-state variables, the machine can have at most 2^p internal states.

All functions (output and next state) will be realized in the same general manner. This will make the internal structure somewhat regular, and allow the modules to be built and then programmed. Inside the chip, $m+p$ combinational functions must be realized where each function can have $n+p$ variables. Additionally, some delay units must be included along with the ability to clock the next-state variables into the delay units. On the chip the delay units will be constructed of clocked flip flops.

Because double-rail logic is not available in ROAM2, no attempt will be made to realize an arbitrary sequential machine using ROAM2. In the ROAM1 realization, each individual function will be realized in a sum of products form. Figure 17 shows the general block

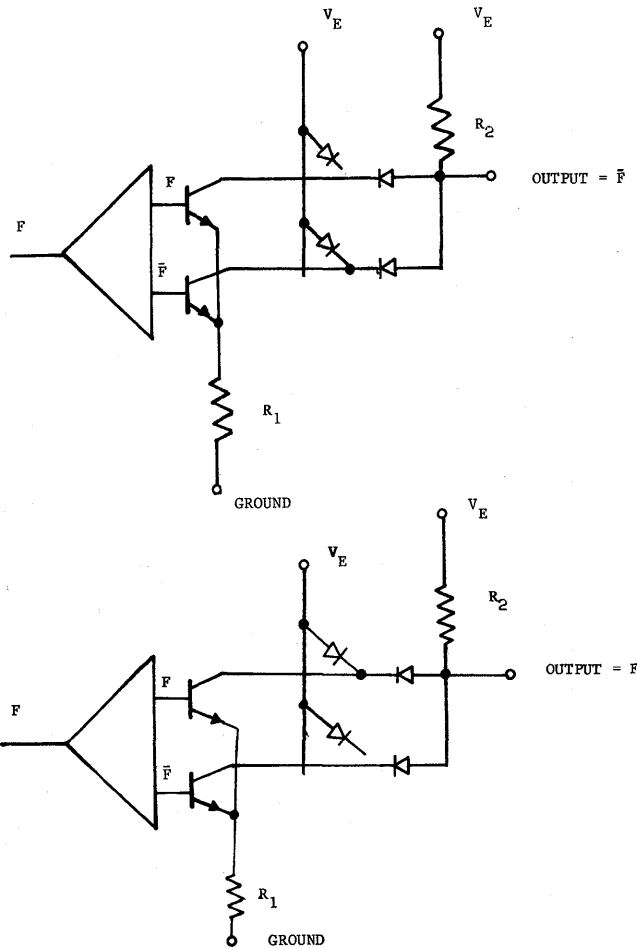


Figure 18—A ROM network for producing for \bar{F}

diagram realization of the system in ROAM. One advantage of the ROAM approach is that no decoders are necessary. Also, using the network shown in Figure 18, a function or its complement may be selected. Therefore, each function of $n+p$ variables requires at most 2^{n+p-1} implicants to realize it. Each implicant requires $n+p$ bits of ROAM1 and each bit of ROAM1 requires the equivalent of two bits of ROM (worst case).

Neglecting flip flops and output drivers (because their cost is about the same for all solutions) a cost of $[c(2^{n+p-1})(n+p)] [m+p]$ bits of ROM is necessary to build the sequential machine from ROAM1 (the function selection ROM described in Figure 18 was neglected). Using a value of two for c [equivalent to selecting a ROAM1 cell that uses two bits of ROM as shown in Figure 3(c)], the resultant cost of the machine is $(2^{n+p})(n+p)(m+p)$ bits of ROM. Since the cost of such an arbitrary machine is a function of three variables the cost equation was evaluated for several

n is equal to 2

		m				
		1	2	3	4	5
p	1	48	72	96	120	144
	2	192	256	320	384	448
	3	640	800	960	1120	1280
	4	1920	2304	2688	3072	3456
	5	5376	6272	7168	8064	8960
	6	14336	16384			

n is equal to 3

		m				
		1	2	3	4	5
p	1	128	192	256	320	384
	2	480	640	800	960	1120
	3	1536	1920	2304	2688	3072
	4	4480	5376	6272	7168	8064
	5	12288	14336	16384		

n is equal to 4

		m				
		1	2	3	4	5
p	1	320	480	640	800	960
	2	1152	1536	1920	2304	2688
	3	3584	4480	5376	6272	7168
	4	10240	12288	14336	16384	

n is equal to 5

		m				
		1	2	3	4	5
p	1	768	1152	1536	1920	2304
	2	2688	3584	4480	5376	6272
	3	8192	10240	12288	14336	16384

n is equal to 6

		m				
		1	2	3	4	5
p	1	1792	2688	3584	4480	5376
	2	6144	8192	10240	12288	14336

Figure 19—Sizing guidelines for sequential machines made out of ROAM1

different values of n, m, p subject to certain constraints. It has been assumed that in the near future ROMs with 8192 bits^{1,10,11} will be available. Since 8192 bit ROMs will be available, it is felt that ROAMs of 16,384 equivalent bits of ROM will be available since

n is equal to 2

		m				
		1	2	3	4	5
p	1	16	24	32	40	48
	2	48	64	80	96	112
	3	128	160	192	224	256
	4	320	384	448	512	576
	5	768	896	1024	1152	1280
	6	1792	2048	2304	2560	2816
	7	4096	4608	5120	5632	6144

n is equal to 3

		m				
		1	2	3	4	5
p	1	32	48	64	80	96
	2	96	128	160	192	224
	3	256	320	384	448	512
	4	640	768	896	1024	1152
	5	1536	1792	2048	2304	2560
	6	3584	4096	4608	5120	5632
	7	8192				

n is equal to 4

		m				
		1	2	3	4	5
p	1	64	96	128	160	192
	2	192	256	320	384	448
	3	512	640	768	896	1024
	4	1280	1536	1792	2048	2304
	5	3072	3584	4096	4608	5120
	6	7168	8192			

n is equal to 5

		m				
		1	2	3	4	5
p	1	128	192	256	320	384
	2	384	512	640	768	896
	3	1024	1280	1536	1792	2048
	4	2560	3072	3584	4096	4608
	5	6144	7168	8192		

n is equal to 6

		m				
		1	2	3	4	5
p	1	256	384	512	640	768
	2	768	1024	1280	1536	1792
	3	2048	2560	3072	3584	4096
	4	5120	6144	7168	8192	

ROAM1 does not need decoders and therefore has more area for bits. Using these bounds, Figure 19 shows the number of bits of ROM necessary to construct an arbitrary sequential chip with the designated values of n, m, p . The values are only calculated until 16,384 bits are reached. In interpreting the figure, n is the number of external input variables, m is the number of output lines, and p is the number of feedback variables (therefore, a 2^p state machine can be realized). The number in row p of column m of the table for a specific value of n is the number of equivalent bits of ROM required.

In realizing the machine using only ROM, the cost was computed to be $(2^{n+p})(m+p)$. Neglecting the decoder (this is done by allowing only 8192 bits), the output drive circuitry and flip flops, the cost of realizing one function is 2^{n+p} and therefore since $m+p$ functions must be realized, a resultant cost of $(2^{n+p})(m+p)$ is obtained. This can be compared to the cost of the ROAM1 solution since the same factors were neglected in both cases. Comparing Figure 20 to Figure 19 it can readily be seen that ROM is better suited for the realization of an arbitrary sequential machine.

A GENERALIZED LOGIC ARRAY

In the preceding sections ROMs and ROAMs were used to solve three different problems. As can be seen from the solutions given, there seems to be no universal array applicable to all types of logic problems. In addition for any given problem either ROM or ROAM seems to fit the problem, but not necessarily both. It was the author's intent to give a detailed illustration of the power of memory (particularly read only semiconductor memory) used as logic and to determine design guidelines for their use; however, there is an additional array that can be made from read only memories. Since this memory is an extension of both the ROM and ROAM it is felt that this array should be introduced and an example given which illustrates that it too has a use. In the preceding sections ROAM was sometimes used for decoding purposes. The GLA is an extension of this concept.

Figure 21(a) shows the Generalized Logic Array (GLA). The GLA consists of a combination of ROM and ROAM. From the block diagram of the array shown in Figure 21(a), it can be seen that this array can be specialized to be either a ROM or a ROAM as previously examined. In order to obtain a ROM (ROAM), the ROAM (ROM) is not used in the array. This array, as shown, combines the best properties of both types of memories; i.e., the table look up (associative property) of the ROAM and the random access properties of the ROM. The combination of associative and random access memories has been used

Figure 20—Sizing guidelines for sequential machines made out of ROM

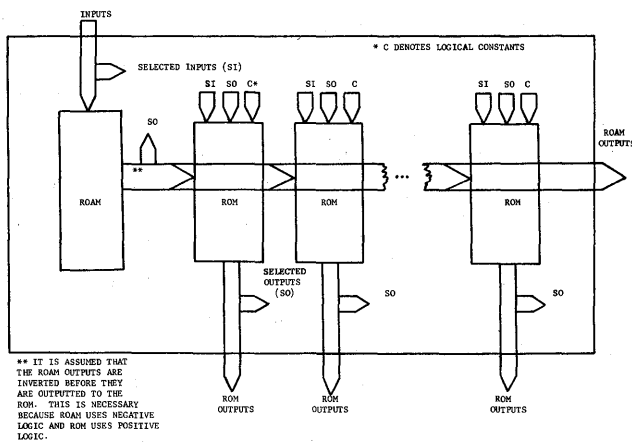


Figure 21(a)—Block diagram of the GLA

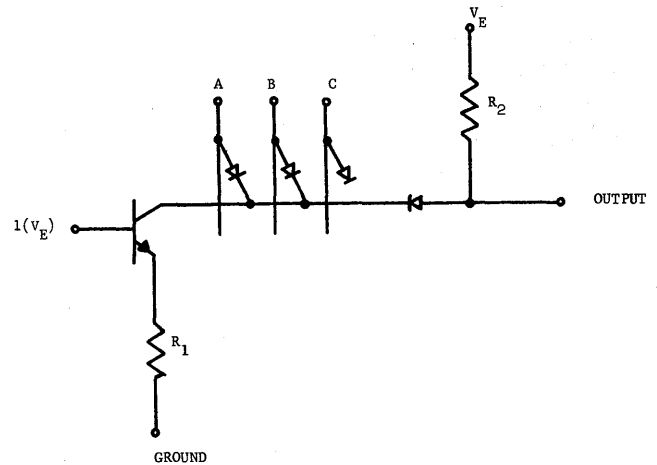


Figure 21(c)—The use of ROM in the GLA to select the appropriate inputs to be ORed together

to advantage previously,¹² but for a different purpose and in a different manner.

This array could be applied to the design of a universal combinational logic module as follows:

- for a ROAM1 implementation delete all ROM and make the ROAM from ROAM1.
- for a ROAM2 implementation delete all ROM and make the ROAM from ROAM2.
- for a ROM implementation use the ROAM as a decoder and the ROM to produce functions that are most economically implemented in ROM.

This array encompasses both ROM and ROAM and

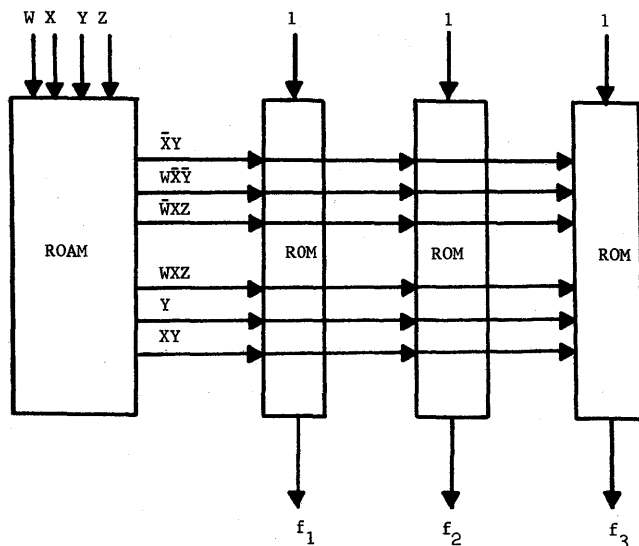


Figure 21(b)—Realization of multiple output function in the GLA

has as its main benefits the following:

- Combines the best properties of ROM and ROAM.
- Allows the implementation of functions which require both associative and random access addressing.
- Is more universal than either ROM or ROAM.
- Limits random wiring on the chip to be external to the memory areas, thus simplifying the layout task.
- Retains a structure in the sense that the ROAM and ROM sections can be considered as macro cells.

The GLA does have a drawback and it is that the logic designer is going to be required to be more ingenious than before.

The judicious combination of random access and associative memories seems to be quite promising for certain types of problems. One of these is illustrated below.

The problem to be solved here is the implementation of the three functions f_1 , f_2 , and f_3 . This problem is taken from page 159 of McCluskey's book.⁹ The functions are defined as follows:

$$f_1(W, X, Y, Z) = \sum(2, 3, 5, 7, 8, 9, 10, 11, 13, 15)$$

$$f_2(W, X, Y, Z) = \sum(2, 3, 5, 6, 7, 10, 11, 14, 15)$$

$$f_3(W, X, Y, Z) = \sum(6, 7, 8, 9, 13, 14, 15)$$

The minimal sums for these three functions are as follows: (page 164 of Reference 9)

$$f_1 = \bar{X}Y + W\bar{X}\bar{Y} + \bar{W}XZ + WXZ$$

$$f_2 = \bar{W}XZ + Y$$

$$f_3 = W\bar{X}\bar{Y} + XY + WXZ$$

Realizing the three functions in ROM would require three 16-bit ROMs plus decoding. The total estimate (in ROM bits) for a realization of this function is 96 bits (allowing the equivalent of 48 ROM bits for decoding). A ROAM1 implementation would require the implementation of nine implicants (you cannot wire OR the $WXZ(\bar{W}XZ)$ term into both f_1 and f_3 (f_1 and f_2) thereby causing you to have to realize all nine implicants and wire OR them only in their specific function) at a cost of eight ROM bits per implicant for a total cost of 72 ROM bits. ROAM2 has a slightly cheaper realization because \bar{Z} does not appear anywhere. Therefore if a seven variable ROAM2 is available and the variables are assigned as $X, \bar{X}, Y, \bar{Y}, Z, \bar{W}$, and \bar{W} , then ROAM2 requires 9×7 or 63 ROM bits for the realization. In this problem it appears that ROAM2 is the cheapest solution with a cost of 63 and ROM is the most expensive with a cost of 96; however, a GLA solution using ROAM2 (ROAM1) can be given that requires only 60 (66) equivalent ROM bits.

This implementation is shown in Figure 21(b). This implementation is based upon the following observations:

- $A = A + O, O = OX$
- there are six distinct multiple output prime implicants: $\bar{X}Y, W\bar{X}\bar{Y}, \bar{W}XZ, WXZ, Y,$ and XY .
- $f_1 = \bar{X}Y + W\bar{X}\bar{Y} + \bar{W}XZ + WXZ + OY + OXY$
 $= \bar{X}Y + W\bar{X}\bar{Y} + \bar{W}XZ + WXZ$
- $f_2 = \bar{W}XZ + Y + O\bar{X}Y + OW\bar{X}\bar{Y} + OWXZ + OXY$
 $= \bar{W}XZ + Y$
- $f_3 = W\bar{X}\bar{Y} + XY + WXZ + O\bar{W}XZ + O\bar{X}Y + OY$
 $= W\bar{X}\bar{Y} + XY + WXZ$

In Figure 21(b) each of the six multiple output prime implicants have been realized using ROAM2 (ROAM1) at a cost of 42 (48) equivalent ROM bits. Decoding has been eliminated (the ROAM really decodes into the multiple output prime implicants) and the three ROMs used cost six bits each for the total cost of 60 (66) using ROAM2 (ROAM1). The three ROMs are 6 by 1 ROMs and contain a connected diode to select the proper terms of the function. The seventh line into each ROM is always kept at logical 1, thus allowing the output of the ROAM to select the output value. For f_1 the diodes driven by the Y and XY outputs are just not connected thereby yielding the correct realization for f_1 . The other two functions are similarly realized. In this example the ROM just serves to mask certain values and to "OR" the proper implicants together to realize the functions. This operation may be illustrated by the three bit ROM shown in Figure 21(c) which produces $A+B$. The multiple

output prime implicants would drive the ROM at points equivalent to points $A, B,$ and C of Figure 21(c). Using the ROM in this manner, the ROM effectively selects the proper outputs of the ROAM and ORs them together if the diode is connected to both lines.

SPEED CONSIDERATIONS

Since the modules have not been built, there is no way to accurately measure the modules' speed. The speed projections given here are based upon speed projections for ROMs.^{2,3,4}

It is important to note that all of the modules are just read only memory so that as far as a module itself, there are no real internal gate delays. For MOS the general sequential machine should be able to run with data changing every millisecond or less. There are some MOS read only memories² that have access times on the order of 50 nanoseconds. Although these access times are not necessarily for a MOS memory of 8000 bits, the general machine running asynchronously could possibly run significantly faster than 1 MHz. If it is running synchronously, it would probably have to be slower (on the order of 1,250 nanoseconds¹⁵). Of all the modules, the general sequential machine is probably the slowest, however, it is anticipated that this module will probably furnish the fastest overall system that could be put together. The main speed advantage the general sequential machine has is that its speed is dependent only on the clock frequency and does not depend upon the number of gate delays that the signals must propagate through as in a random logic implementation.

It is anticipated that the sequential logic module will be able to operate at a 600 nanosecond access time in MOS¹⁰ and a 35 to 70 nanosecond access¹³ time in bipolar implementations. These speeds seem reasonable and compare favorably with logic that is available today; however, it should be anticipated that these speeds will be heavily dependent upon the use and design of the clocked flip flops. Again, the device is really only a read only memory and can be operated as fast as a read only memory. There are available ROMs which can run a lot faster than those assumed here^{13,16} and ROM speeds are projected to improve significantly.²

Evaluation of the speed of the universal combinational logic module is fairly straightforward since it does not contain any internal feedback loops with flip flops in them. In a MOS implementation, one would expect currently to be able to operate the module around 2 MHz, regardless of whether you are changing the module's function. A bipolar implementation should be able to operate at 20 MHz or above. Both of these

anticipated speeds are below the projected speeds for read only memory because there is some internal feedback on the flip-flop functions. It would be better to operate the module slower than capacity to assure that the output values are correct. The MOS speed (0.5 millisecond) is slower than most comparable MOS logic (typical speeds for the Electronic Arrays 1800 Series range from 250 nanoseconds to 150 nanoseconds). The logic building blocks most comparable to the universal combinational building block are the EA1800 (250 nanoseconds) and the EA1806 (250 nanoseconds). However, ROM speeds are expected to increase and are increasing on smaller ROMs.¹⁶

CONCLUSION

The viability of read only memories for use as logic devices was investigated in this paper. Three different versions of read only memory were applied to three different problems. The three problems that were considered are: (1) use of read only memories to build a universal combinational logic module to replace a large number of ICs and which could be used to generate random logic functions, (2) use of read only memories to build a sequential logic module to replace a number of ICs that could be used to generate subsystems, and (3) design guidelines for constructing a sequential subsystem on a single chip. A new type of read only memory was introduced which incorporates the best features of ROM and ROAM.

The three types of read only memories that were applied to the problems are ROM, ROAM1, and ROAM2. Each different type of read only memory has its own particular properties that tend to make it quite unique when compared to the other devices. ROAM2, in general, does not have the ability to perform double-rail logic. This makes ROAM2 particularly efficient at realizing simple logic functions such as AND. ROAM1 has double-rail logic and can therefore implement very complex functions; however, it requires a large amount of memory to realize a function unless the function can be simplified to a small number of minterms and implicants. ROM has the advantage that it takes the same number of bits to realize any function of n variables, regardless of complexity. The hybrid read only memory GLA was also applied to a problem and was shown to be able to synthesize multiple output functions effectively.

Applying the three different types of devices to the three problems showed that each device has its place and no device is optimum for all problems. ROAM2 was the best choice for the universal combinational

logic module. This was true because the functions in the module could be realized using mostly AND logic. ROAM1 was best suited for the sequential logic module (ROAM2 was not considered because it generally doesn't have double-rail logic). This was true because the functions considered were able to be simplified to a small number of implicants, whereas ROM was best suited for realizing the arbitrary sequential circuit on a chip. The best fit for any problem will probably be found by specializing the GLA into either ROM, ROAM, or a hybrid of ROM and ROAM.

The results of this study indicate that all types of read only memories are necessary for efficient realization of functions. In this study each different function used a different type of read only memory for its optimal realization. Depending on the properties of the system to be realized, different types of read only memory may require a different number of bits (in the universal combinational logic module, the difference was over an order-of-magnitude between a ROAM2 realization and a ROM realization).

The following guidelines for selection of memory type were obtained during this study:

1. Size the problem for GLA and the appropriate guideline from two, three, and four.
2. If double-rail logic is not required and $I < 2^{(n+1)}/n$ use ROAM2.
3. If double-rail logic is required and $I < 2^n/n$ use ROAM1.
4. If none of the above use ROM.

where

I = number of implicants

n = number of variables.

Some of the major benefits offered by the use of read only devices are that whole families of logic may be replaced by one read only memory chip that is programable, thereby increasing the demand for this device and lowering costs, and the development of read only chips is inexpensive enough that companies can develop their own ULM and not be restricted to using logic defined by semiconductor manufacturers. In addition, read only memories are well suited for large volume production and testing. Other major benefits are as follows: (1) it may be possible to design a multi-technology compatible device by having different power supply pins available where each pin utilizes different bias supply resistors to obtain different output voltages; thereby obtaining a device capable of driving different technologies based upon the supply

pin chosen; (2) low development cost compared to custom-designed LSI chips; (3) lower inventory necessary for replacement and repair purposes; (4) short development lead time in comparison to the development lead time for a LSI chip; (5) regular logic layout so that yields can be high and device density maximized; (6) very little random logic on the chip; (7) fast, and (8) programmable to produce a number of functions.

It is felt that read only memories do offer a solution to the LSI random logic dilemma and that read only memory can be very useful in performing logic functions in systems, but that all types of read only memory are necessary and no one type of read only memory can solve all problems in the future unless it is a hybrid such as the GLA.

APPENDIX

DESCRIPTION OF HOW ROM OPERATES

Figure 1(a) shows a 16-word by 1-bit read only memory (ROM). This device is operated by placing a four-bit address onto the four input lines $I_1, I_2, I_3,$ and I_4 . This address then reads out the bit stored at the corresponding address in the ROM. If the diode at the address is connected to both lines a one (V_E) is read out, otherwise a zero (ground) is read out.

To see how the device actually works, assume that $V_E > \text{ground } (G)$ and $R_2 \gg R_1$. When a four-bit address is placed on I_1, I_2, I_3 and I_4 exactly one of the lines $A, B, C,$ and D becomes V_E and the rest are G . Simultaneously, exactly one of lines one, two, three and four becomes V_E and the other lines are G . Assume lines C and two are selected and become V_E . The transistor T_c then forms a path for current to flow through R_1 to ground. This current is supplied by one of two sources. If a diode exists at the intersection of lines C and two [as it does in Figure 1(a)] then the current is supplied through the diode at the intersection of lines C and two; i.e., point 0_c is at V_E which forces the output to be near V_E .

In order to see what happens when the diode is not connected, assume a new address is placed on the input lines and lines two and B go to V_E . The diode at the intersection of lines two and B is not connected; therefore, point 0_B tends to go to G and current is supplied by the power supply through R_2 and D_B to T_B which is a path to ground. Since R_2 was made much larger than R_1 , the output is forced to G . The diodes $D_A, D_B, D_C,$ and D_D prevent unwanted currents from flowing. In the case under consideration, lines two and B are at V_E , but since line two is at V_E , the diode at the inter-

section of lines two and C wants to form a current path. Since T_c is not turned on (line C is at G), current would tend to flow to point 0_c , then to point 0_B (raising the output to V_E) finally to ground through T_B if diode D_c were removed. Therefore, diode D_c is necessary to stop this unwanted flow of current and keep the output at ground (G) which is what was desired.

DESCRIPTION OF HOW ROAM WORKS

Figure 3(a) shows a three-input, one-output, read only associative memory (ROAM). This device is operated by placing a three-bit address onto the three input lines $A, B,$ and C . If this address matches an address that has been previously stored in the ROAM, then a one appears on the output line of the ROAM. If the address doesn't match any of the addresses stored in the ROAM a zero appears at the output of the ROAM. This network can be used to generate logic functions by associating a minterm with a minterm that has been previously stored in the ROAM.

To see how the circuit actually works, assume that $-V_E < \text{ground } (G)$ and that $R_2 \gg R_1$. If negative logic is assumed ($-V_E = 1$ and $G = 0$) then the circuit in Figure 3(a) can be used to perform the logic function $f = A\bar{B}\bar{C} + \bar{A}C + ABC + \bar{A}\bar{B}\bar{C}$. Assume that $A = 1, B = 1,$ and $C = 1$ is put onto the input leads then in row 3 of the ROAM in Figure 3(a), $-V_E$ appears at the vertical connection point of all diodes in row 3. This leaves point 0_3 near $-V_E$ and current flows through $R_2, 0_3,$ and R_1 to the terminal at $-V_E$. Since $R_2 \gg R_1$, the output is about $-V_E$ and a logical 1 appears at the output. All other rows are mismatched and each of the lines $0_1, 0_2,$ and 0_4 were held to ground because (considering row 1) in each row at least one diode's anode was tied to ground, causing the horizontal line to be at ground, therefore, not biasing the diodes D_1, D_2 and D_4 on. If the address was a mismatch in all rows, $0_1, 0_2, 0_3,$ and 0_4 would all be near G and the output would be G .

In summary, a mismatch on any bit causes the row to leave the output at ground. Any row that has all matches ($-V_E$ on the anode of all connected diodes in that row) will allow the output diode of the row to conduct pulling the output negative (logical 1). One advantage of ROAM is that "don't care" conditions can be programmed in by not connecting diodes. If positive logic were used and $G = V_E$ and $-V_E = 0$ then logic functions compatible with ROM could be realized because the ROAM would then produce OR-AND logic. In positive logic, the ROAM shown in Figure 3(a) produces the function

$$f = (A + \bar{B} + \bar{C})(\bar{A} + C)(A + B + C)(\bar{A} + \bar{B} + \bar{C})$$

REFERENCES

- 1 *MOS firm puts 4096-bit memory on a single chip*
Electronic News p 1 January 19 1970
- 2 *Designer's guide: Semiconductor memories*
EEE pp 53-67 November 1969
- 3 *Semiconductor memories Part III, Biopolar RAMs and ROMs*
Electronic Products p 23-25 March 1970
- 4 R F GRAHAM M E HOFF
Why semiconductor memories
Electronic Products pp 28-34 January 1970
- 5 J L NICHOLS
A logical next step for ROM
Electronics pp 111-113 June 12 1967
- 6 J C LEININGER
The use of read-only storage modules to perform complex logic functions
International Computer Group Conference Washington DC June 1970
- 7 *Microcircuits, IC complexity due to double yearly*
Electronic Design pp U93-U97 March 15 1970
- 8 R A HENLE et al
Structured logic
AFIPS Conference Proceedings Fall Joint Computer Conference 1969 pp 61-68 November 1969
- 9 E J McCLUSKEY
Introduction to the theory of switching circuits
McGraw-Hill New York
- 10 *4096-bit MOS/LSI*
Electronic Products Magazine p 57 June 21 1971
- 11 *Most complex ROM shrinks CPU*
EDN pp 16-17 June 15 1970
- 12 K J THURBER
An associative processor for air traffic control
1971 SJCC Proceedings AFIPS Press Volume 38 pp 49-59 May 1971
- 13 *Biopolar 1024-bit ROM accesses in 30 ns*
Electronic Design p 64 March 4 1971
- 14 *High-density 5120-bit ROMs include on-chip decoding*
Electronic Design p 95 November 8 1970
- 15 *Specification sheet for the EA 3307 ASCII/EBCDIC code converter ROM*
Electronic Arrays Inc
- 16 *2560-bit memory has 500-ns access*
Electronic Design p 95 November 8 1970
- 17 W I FLETCHER A M DESPAIN
Simplify combinational logic circuits
Electronic Design pp 72-73 June 24 1971
- 18 J WUNNER R COLINO
Applying the versatile MOS ROM
Electronic Products pp 35-40 January 1970
- 19 H SCHMID D BUSCH
Generate functions from discrete data
Electronic Design pp 42-47 September 27 1970
- 20 *Programmable logic arrays*
Texas Instruments Bulletin CB-126 pp 152-166 October 1970

A panel session—Computers in medicine—Problems and perspectives

Medical Information Systems: A Reformulation of the Problems as Perceived by a Hospital Administrator

by BALDWIN G. LAMSON
UCLA Hospitals and Clinics
Los Angeles, California

Much has been said in recent years of the inefficiencies of hospital management and patient care, and of the opportunities for automating the recording of nurses' notes, scheduling of drug administration, recording of doctors' orders, dietary menu planning, diagnosis by computer, and the like. The hospital business office and revenue accounting have often been singled out as the only areas in hospitals where modern data processing equipment has been at all effectively used.

After a decade of effort to produce total hospital management and communication systems, few, if any, completely successful and self-sustaining systems are in full operation. Many of the major problems still remaining which to date have defied complete solution were noted ten years ago but are now recognized as being vastly more complex and difficult of solution than originally perceived.

The medical record remains the heart of the problem. Hard data, such as clinical laboratory reports, have been successfully processed by computer, but doctors' observations and physical examination data still largely defy satisfactory input solutions. Patient self-query systems have proven practical and are anticipated to come into wider use.

The organization of the medical record for most efficient computer storage and retrieval is still a major challenge. Much of the medical record as it accumulates is obsolete within forty-eight hours, except for medical/legal and medical research purposes. Very possibly, promptly dictated and typed uncoded records of day-to-day observations of the physician, followed by a single well-structured summary and analysis upon the conclusion of each episode of medical care, will suffice.

Systems for scheduling of patient appointments for patient convenience and proper utilization of facilities remain a very high priority. This problem is solvable and awaits only adequate capital and a dedicated effort.

Despite past successes, the fiscal system is still desperately in need of further assistance because of the complexities of federal and state health care legislation. A national effort is needed to make available to participating hospitals up-to-date information with respect to each patient's prior utilization of health facilities, status of deductibles and coinsurance, as well as location and content of prior medical records. The challenge here is more political than technical.

The goal of a completely integrated hospital computer-based information storage and communication system is lower in priority. Partial stand-alone applications appear more attainable and are urgently needed. Problems related to patient identification, patient scheduling, medical record location control, and patient eligibility are most urgent.

Technology already appears to be adequate to do the things that are needed most. Slow progress is more related to the absence of research and development capital in the hospital industry. The short term opportunities appear greater for software development than for equipment, although techniques for the rapid retrieval and transmittal of full page facsimilies by video screen with optional hard copy output would appear to have a large market in the health service industry.

New Technologies in Medicine

by C. T. POST, JR.

Department of Health, Education, and Welfare
Rockville, Maryland

We are at the threshold of a technological era in medicine. The Government is interested in harnessing technology to alleviate manpower, cost, access and distribution problems existing in the health care system today. There exist, on the one hand, several examples of technology in use today which clearly improve the quality of medical care but which do so at considerable cost. On the other hand the techniques which have the potential for reducing the cost and increasing the availability of health services can only do so when they are

deployed in situations where sufficiently large populations can be aggregated to take advantage of the economies of scale that are implicit in these techniques. There is a pressing need to demonstrate that these potentialities for economy and improved access can be realized. This can only be achieved by mounting fairly extensive experiments that will make evident the economies of scale that are inherent in these technologies. This type of demonstration is made feasible by readily available communication capabilities which now exist. The fact that it is now possible for a large number of medical institutions to share a common automated medical service has very clear and immediate implications not only for the reduction of unit costs but also for the dissemination of high quality medical information, advice and services. In addition, the sharing of this service will exert pressures that will move communities of hospitals toward the sharing of other services and toward standardization of their operations. Both within the hospital, and to a greater extent within doctors' offices, the necessity for development of modular, evolutionary, user-oriented reliable systems is becoming increasingly evident. This latter area, the physician's office, where the majority of medical care is in fact delivered in this country today, remains largely untouched by technological aids. Given the persistence to a large extent of the present organizational scheme of primary care delivery, successful entrants into this marketplace will in all likelihood primarily operate within the physician's environment to tap his thinking for purposes of problem definition, and then reconfigure existing technological components into a purely problem solving user-oriented system.

The Role of Computers and Information Systems in Medicine

by E. E. VAN BRUNT

The Permanente Medical Group
Oakland, California

An information system can be defined as a system intended to provide information needed by the user in

the conduct of his business. The primary 'business' of medicine is the care (management) of people who require various levels of medical investigation, counseling and treatment. While many specialized medical and medical administrative data processing capabilities have been developed, the current systems of medical information management and communication are inadequate for present day patient care needs.

Medical care is comprised of both hospital and clinic activities; the larger component exists in the outpatient care areas. In this medical center environment, the heart of any information system is an integrated, or continuous, lifetime record—for each of the patients receiving medical services, and the data system—manual and/or electronic, which supports its growth, maintenance and utilization. Concomitant with the progressive development of group or 'regional' medical care programs is the need for more effective large volume medical information management. The objective is the timely supply of relevant information to appropriate users, for patient care services, and medically-oriented research and education.

The supply of information supporting patient care implies extensive and highly reliable communication capabilities: the communication of patient data from the professional providers of care to the medical record, and to other professionals and service bureaus; communication, on demand, of relevant summary information from the patient's record to medical professionals and service bureaus; communication between services. The conduct of medically-oriented research implies existence of a medical data base that can support clinical, epidemiological and health services research. This same data base should support limited, patient care oriented, educational services to medical care professionals.

The role of computer-supported information systems in the medical care environment is clear but outstanding problems exist in both the medical and computer-oriented disciplines.

*Medical data system studies have been supported in part by a National Center for Health Services Research and Development (NCHSR&D) Grant (HS-00288) and by the Kaiser Foundation Research Institute.

A panel session—The user interface for interactive search

The User Interface for Interactive Search

by JOHN L. BENNETT

IBM Research Laboratory
San Jose, California

In January 1971, the AFIPS Information Systems Committee sponsored a workshop on "The User Interface for Interactive Search of Bibliographic Data Bases." This narrower topic was chosen intentionally to give the Workshop a focus suitable for intensive discussion within a small group. Now that the Proceedings are available it is appropriate to highlight those user interface characteristics shared by a less restricted range of applications. The keynote for the panel will be "what goes on in front of the terminal"—what facilities the user requires, what services the computer can provide, and how the user responds to data display. Understanding the exchange of data between the user and the computer at the interface during search will enable designers to implement systems truly responsive to search needs.

Each member of the panel will relate his own experiences to the keynote subject of interactive search. Bennett will draw on work with the Negotiated Search Facility which was used to study search behavior given the data of bibliographic files. Walker, as editor of the

Proceedings, will comment on Workshop accomplishments and make observations based on the SHOEBOX personal file system developed at MITRE. Engelbart, an innovator in the use of terminals, will describe what has been learned from experiments at SRI which place the full power of the computer at the service of the user. Katter has examined a variety of interactive applications at SDC, and has developed a model of user behavior at the search interface which illuminates problems and suggests a direction for their solution. Hugo will comment on the service to be provided for a large class of noncaptive users—Senators, Congressmen, and their administrative staffs—for whom interactive access to a data base would represent just one of a set of information-gathering tools. Morton will draw on studies made at MIT and Westinghouse to tell us what interface facilities are needed to establish the conditions under which business decision makers would conduct their own searches rather than delegating terminal interaction to others.

In our discussion, we will identify those findings currently available which help us design the link between the user who understands the search results he wants and the system designer who can provide the means for achieving those results. Though the emerging interface technology is only roughly defined, we can begin to outline now the research and development issues to be resolved if interactive search is to achieve widespread user acceptance.

A panel session—State of the computer art in biology

Computer Applications in Cellular Research—Three-Dimensional Brain Reconstructions

by CYRUS LEVINTHAL

Columbia University
New York, New York

Computer applications in cellular research range from relatively straightforward data reduction to complex modeling. Programs exist for interpreting Coulter counter data, for assisting microscopic cytological assessments when fluorescent antibodies are used, and for relating observed uptake of labels to cell-cycle parameters. Models of cellular populations range from simple cycling systems supportive to data reduction (e.g. for label uptake) to complex models incorporating feedbacks and cellular differentiation, which are being used to explore better strategies for treating leukemia and cancer. The greatest need is for better experimental information. Programs enabling reliable automatic analysis of bone-marrow sections or smears probably will be difficult to develop but of great value to experimental hematology. Programs to reconstruct three-dimensional anatomies of cellular systems should provide valuable insights and, as indicated in the following report of current research, are feasible now.

The brains of small organisms can be cut into thin serial sections, each of which can then be photographed in an electron microscope. All information as to the nerve branching patterns and connectivity is contained in this set of photographs, but even for a very simple organism the number of photographs required will be several hundred. Thus, reconstructing the three-dimensional (3-D) information in a usable form is a formidable problem. We have developed a method of combining the photographs in a motion picture film strip in such a way that each section is aligned with the one before it. When the movie is projected, the observer has the illusion that he is traveling through the brain.

Recording of the nerve net is done by superimposing the projected image from the movie with the image on a computer-driven oscilloscope display. During record-

ing, a cursor on the scope is controlled by a hand-held device (a "mouse") with two potentiometers to determine x and y . The movie projector is controlled by the computer, and the frame number is proportional to the z coordinate. Thus, the observer can use the system as a three-dimensional notebook. The cell bodies and branching pattern of fibres, as well as synapse locations, are recorded for each nerve. After all nerves have been recorded a BRAIN file can be constructed by a program which matches the synapses which were separately recorded in each NERVE file.

Any combination of nerves can be displayed in 3-D by rotation of the projected image. Similarity of branching patterns in two different organisms can be determined by matching them analytically with a simple graph-theoretic algorithm or by simultaneous display of nerves from different organisms. In the same way, bilateral symmetry can be identified in individual organisms.

In simple invertebrates having brains with only a few hundred cells but a very complicated set of fibres connecting them, no differences have been observed between two genetically identical organisms. Within an individual organism there is a remarkable degree of bilateral symmetry. A four-dimensional description of a simple brain is now being developed by carrying out the 3-D mapping at various stages during embryologic development.

Computer Applications in Population Studies

by NEWTON E. MORTON

University of Hawaii
Honolulu, Hawaii

Population genetics is the study of forces that change or maintain genotypic frequencies. With support from

computers, research emphasis recently has shifted from the conceptual synthesis of simple models to issues related to the maintenance of genetic variability in actual populations. That man himself has become the preferred organism for many types of population research results not only from motivations for human benefit but also from the fact that the large number of recognizable single-gene differences in man far exceeds the known polymorphisms in *Drosophila* and other classical genetic material. With this research emphasis on human populations, computers become indispensable both for data analysis and modeling.

Human data rarely can be acquired under the well-defined, controlled conditions possible for the laboratory geneticist. Efficient data management is essential in the study of large populations for whom interrelated pedigrees and migratory traces must be maintained. Complex data analyses are required. Improvement of these approaches is an important area for computer implemented biostatistical research. Human data tend to be expensive to acquire and often cannot be replicated; all possible information must be extracted.

Theoretical population models become complex as they are applied to the description of realistic systems. Computer implementation is essential. Efficient modeling techniques must be developed, as well as more effective displays or interactive approaches for model exploration.

Models have been proposed as alternatives to the single-locus Mendelian models for the familial clustering of some congenital malformations and many of our more common diseases, e.g. club foot, pyloric stenosis and diabetes. The major contribution of these models was realized only after they were extended and programmed for the computer. This additional complexity permitted the direct computation of genetic risk figures. In the near future, it will be practical to define upper and lower boundaries on the risk under a wide variety of situations, permitting much more direct genetic counseling. Using the computer to extend these models and to apply them to a large amount of data is continuing to stimulate the development of new approaches and modifications in the underlying theory.

Another important area for stimulating interaction between computer advances and population research is the problem of record linkage. Testing through application of population models often requires tremendous data sets available only by accessing multiple large files such as of birth, marriage, and death certificates. With the development of more efficient and accurate management of these files, important aspects of hypothesis testing may be pursued and continuing theory and model development is stimulated.

Molecular Biology as it Relates to Digital Image Processing

by ROBERT NATHAN

California Institute of Technology
Los Angeles, California

Biological research has been placing increased attention upon the determination of the atomic structures of large molecules. These have generally been proteins and especially enzymes whose molecular weights range up to 50,000. Tremendous efforts to infer functional configurations of active enzyme sites have been expended by organic chemists. But total atomic configuration has been performed exclusively by the digital computer using the methods of x-ray crystallography. The sequences of the amino acids in these enzymes have been determined by organic chemists. The information is then used by the crystallographer to infer initial estimates about the actual geometrical configuration, which the crystallographic data eventually confirm. Several of these molecules and their functions will be described.

There are many other large biological molecules and molecular aggregates whose structures should be determined if we are to further unravel the mysteries of cell function and apply these solutions to the problems of disease. What are the structures of *t*-RNA, ribosomes, mitochondria, membranes, antibodies, and even whole viruses?

In our laboratory, several computer techniques have been developed to manipulate continuous-tone digitized images. These methods were originally applied to space photography. They are now being applied to medical x-ray images, and to light and electron micrographs.

(A brief description of automated karyotyping, the light-microscope analysis of chromosomes, is included as an example of light-microscope computer automation.)

An example of micrograph enhancement of the normal electron microscopy of the enzyme catalase is shown to illustrate present microscope limitations.

The final discussion is centered around a description of a method for computer manipulation of dark-field electron micrographs which should eventually reveal atomic structure without the assistance of chemical inference. A crystal of an organic dye, indanthrene olive (molecular weight 750) is chosen to illustrate the computer method of obtaining high resolution by means of a system called synthetic aperture. A preliminary atomic resolution model is presented.

Automated Information-Handling in Pharmacology Research

by WILLIAM F. RAUB

National Institutes of Health
Bethesda, Maryland

Pharmacology involves the multitude of interrelationships between chemical substances and the function of living systems. Since these interrelationships manifest themselves at all levels of physiological organization from the individual enzyme to the intact mammal, research in this area involves concepts and techniques from almost every biomedical discipline. Thus, pharmacology entails a class of information-handling problems as formidable and enticing as any that can be found in the medical area. In recognition of this, the National Institutes of Health (NIH), through its Chemical/Biological Information-Handling (CBIH) Program, is attempting to accelerate the acquisition of new pharmacological knowledge by designing and developing special computer-based research tools. Working through a tightly interconnected set of contracts with universities, research institutes, profit-making organizations, and government agencies, the CBIH Program seeks to blend the most advanced information science methods into a computer system which can be an almost indispensable logistical and cognitive aid to these investigators.

In the absence of all-encompassing theories of drug action, pharmacologists rely primarily on empiric observations communicated in a plethora of literature with which currently available information-handling systems are unable to cope. There must be an increased emphasis on effective data retrieval, as opposed to document retrieval. Past work on encoding molecular topology has produced two systems, connectivity tables and linear ciphers. The former demand considerable storage and processing time, but the latter exclude the possibility of some important substructure queries. New or combined approaches are required which relate effectively to the task to be performed. A sophisticated man-machine interface is of high priority to effect the interchange of data, procedures, and models among geographically and disciplinarily disjoint scientists whose work is relevant to the understanding of drug action. There exist promising interactive systems for tablet input of two-dimensional chemical graphs and for the graphical display and manipulation of three-dimensional molecular models.

Two projects currently pursued under the CBIH

Program will be discussed:

The PROPHET system is a medium through which the latest pharmacologically relevant information-handling methods can be developed, integrated, and made widely available via a time-shared PDP-10 to practicing scientists whose disciplines range from molecular biology to human clinical investigation. It includes a powerful interactive command language for handling empirical data, a coextensive simple procedural language modeled on PL/I, provisions for easy access to complex computational processes, a rich substrate of devices for handling different kinds of pharmacological data structures, and facilities for communication among users.

Another project is exploring the use of automated inference methods as cognitive aids to pharmacological investigators. At present, model-handling tools are being developed to enable researchers to express and assess the validity of their concepts about mechanisms of drug action. Finite-state automata models have been found especially useful.

Computers in Physiological Modeling

by WILLIAM S. YAMAMOTO

University of California
Los Angeles, California

The principal reasons for the construction of computer models of systemic physiological models are no different from the problem of theoretical computations in any other field. Systemic physiology, by which I mean the written literature of the subject as distinct from the process of acquiring such information, is voluminous, undoubtedly redundant, and probably contains many contradictions. Moreover, it is difficult to separate opinion from observation, correlation from imputations of causality. Nevertheless, this is the milieu on which the intelligent physiologist functions. He pushes the frontier forward in every decreasing salients because the weight of past experience becomes more and more unmanageable.

The following are major reasons for mathematical modeling of physiological systems:

1. Models serve an heuristic role, and can guide laboratory research into unilluminated corners, or applications.

2. Models codify unambiguously extensive systems of postulate or conjecture for purposes of retrieval or ratiocination.

A critique of modeling activities in the light of these purposes finds exact parallels in the laboratory study of physiology. The perspectives of investigators in the laboratory are shrinking. There is specialization because the number of details and procedures as well as the number of plausible alternatives increases with the increase in apparent information. Problems of information synthesis become monumental, and the review article occupies a significant place in scientific literature. Modeling, which I claim is the only testable process of scientific synthesis (review), suffers from the similar problems. Namely, models are not unique, and they become rapidly insensitive to both parameter and structure when a certain level of complexity is reached. The latter implies further that there is for any level of development above some minimum the possibility of alternative but compatible model structures, just as there are alternative but compatible explanations in theories of the same phenomenon.

There have been several models whose scale is sufficient to demonstrate the important problems. These are the model of glucose metabolism, adrenocortical function, the cardiovascular system, and of external respiration. I am sure although such matters are usually not presented in print that all of these investigators have encountered common problems of the following sort. First, one must make a choice between lumping and tabulating the components of parameters. If one lumps, one loses the heuristic value; physiologists can no longer assign biologically meaningful notions to magnitudes. Parameters with common elements become confused. If one does not lump parameters, the glossary keeping and degradation of computation efficiency become progressively severe. In fact, it becomes impractical for the modeller to set aside programs for even periods as brief as two weeks. Second, model sensitivity decreases so that not only parameters but whole sections may be altered in structure and if the test behavior is a limited set, there is no discriminatory function in exercising the model because methods for fitting models to data are not well developed. There is constant need to map model behavior upon the experimental domain,

i.e., just as in the real case, conclusions need to be tested to see if the permutations which constitute laboratory experience can be replicated.

To examine the problems that arise in the translation of physiological ideas into formal models, we chose to add to our model of external respiration in mammals a subroutine which generates the drive for movement of the chest. In the original model this was represented by three statements which produced a hybrid modulated wave in which both the amplitude envelope and the frequency envelope were functions of carbon dioxide concentration in brain cells. The plan is to replace this short subroutine with one which encompasses a substantial part of the recorded observations upon neural mechanisms below the midcollicular level. There are three classes of neurophysiological experiment: ablation, stimulation, and microelectrode recording. Each of these is given an unambiguous symbolic definition. The respiratory complex is then constructed as a net of differential equations in which the principal criteria are location, firing pattern, and chest movement. Fifteen nonlinear first order d.e.'s are used, and synthesis proceeds in reverse of the ablation literature. To produce the phenomenon of gasping merging into apneusis, we are dealing basically with eight of the centers. Apart from parameter size, each center may be connected to any other center in any of 3 ways and the directed graph has potentially $3 \times 7 \times 8 = 168$ patterns not all of which give behavior which is readily discarded. Since one type of connection is the non-connection, there are 112 parameters possible of which at any one time only a few are testable. The computational problem is severe. In the absence of search formalisms the most reliable tool turns out to be a thorough reading of the physiological literature. And the model can be related almost relation by relation (in the code) with statements in the research literature.

For both the educational process and for the improvement of models beyond the easy stage, tools are necessary and they are probably some form of computational tool to handle and demonstrate the structure of other programs, perhaps in graphical form of their relations and their implications. Automatic flow charting is the most primitive form of such a program. If a program's content is a tree or a forest, a topologically sorted list might be a second valuable direction to go.

Introduction to training simulator programming

by D. G. O'CONNOR

Singer-General Precision, Inc.
Binghamton, New York

INTRODUCTION—GENERAL REMARKS ABOUT SIMULATION

Several years ago, while we were having a compiler¹ developed for our use in simulator computer programming, one of the senior management people at our subcontractor's expressed surprise at our dogged insistence on object code "efficiency." Now, in fact, he had not been on this project very long and did not realize that a simulator for our purposes is a product—not a program run on an in-house computer. As I shall interpret the word simulator in the remainder of this discussion, it will mean a product used for training.

What are the implications of this statement? Well, first it implies someone is being trained, and consequently, there is a man in the system someplace. In fact, he interacts quite significantly with the operation of the system. System outputs are cues, and feedback to this man and his actions are inputs to the computer. To be more specific, let us limit our discussion to aircraft^{2,3} and space simulation.

In Figure 1 is shown an artist's sketch of the Lem Mission Simulator (LMS). An appreciation of the size of a complete simulator is derived from this sketch. A sketch is shown rather than a photograph because it is impossible to get a photograph since the physical size of the equipment would constrain a camera to be so far away that it could not be in the same room. In fact, you will note, this equipment is itself in *several rooms*.

Such a simulator consists of several major parts (as seen in Figure 2):

- (a) a training station (cockpit)
- (b) instructor's station
- (c) a computer
- (d) interface equipment
- (e) special effects—e.g., visual and motion equipment.

The training station is a replica of the operating environment, i.e., cockpit, for which the training is being conducted. Frequently, production equipment is used in the simulator, or at least the panels and controls are used. It is fair to say that every effort is made to make the training station environment so close to the real environment that being brought into the environment blindfolded would leave a human with a difficult judgment were he asked whether he was in the real operating location or not. In particular, since we deal primarily with flight simulators, I mean by the foregoing statement that if an experienced pilot were put in the seat of a simulator, he would have difficulty determining whether he were in an airplane or not (except for certain obvious clues).

The instructor's station consists of a console with display of (almost) all the instruments, signal lights, etc., available to the pilot. In addition, other displays may be incorporated, such as CRT display used for instructor assistance and other indicators, lights, and meters of importance to the instructor in performing his job. Controls for FREEZE, MALFUNCTIONS, etc., are located here for the operation of the simulator training problem.

The computer used in such a simulator must be capable of performing "many" calculations per second. The calculational load is determined by the aircraft being simulated and the number of systems aboard this aircraft whose operation must be simulated accurately, as well as by psychological and physiological factors.⁴ In recent years, more and more systems have been simulated and have been simulated to a higher degree of fidelity. This growth has led to requirements for more efficient programs and additional computer capacity. Consequently, significant requirements for improved programming have arisen. In spite of this, increasing requirements have been placed on the computer.

The interface equipment required includes both hy-

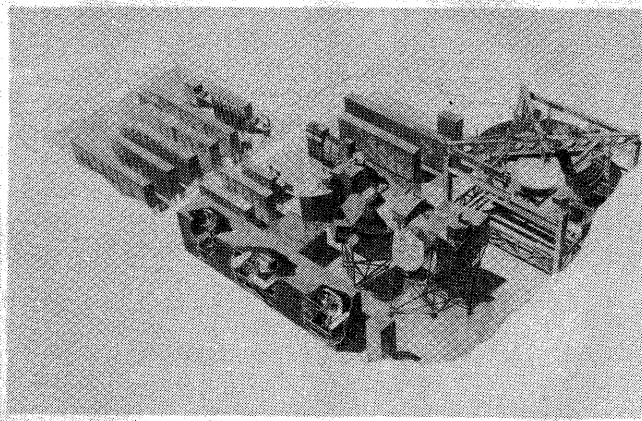


Figure 1—Lem mission simulator

brid (analog to digital and digital to analog conversion) and purely digital inputs and outputs.

The special effects include systems such as a visual system which presents moving scenes to the pilot over a portion of his flight path. Obviously, the scenes presented to the pilot must be consistent with such things as simulated aircraft velocity and attitude as well as simulated position relative to the scenes being portrayed. In some simulators, a device for moving⁵ the cockpit to simulate accelerations is incorporated.

A practical side effect is the almost continuous use to which this equipment is put. The implications of this type of usage are that the computer has the ability to be reloaded and restarted with a minimum of operator activity. Before addressing my topic directly, let me indicate the size of the problem at least broadly.

In Figure 3 is shown a little more detail of the peripheral equipment which may be employed in a flight simulator. Since this equipment is required primarily during program development, there are many simulators delivered with a sharply reduced amount of this equipment.

Also shown in this figure is a list of ranges of some of the parameters pertinent to the computer hardware. The range shown for *Processing Speed* and *Memory* is representative of single or dual processor simulators. Note also the reliability figure. While this figure is remarkable, it has been achieved on many simulators used twenty hours per day, six days per week.

THE SIMULATION PROCESS

The intent of a training simulator is to convince the trainee that he is operating the equipment in reality.

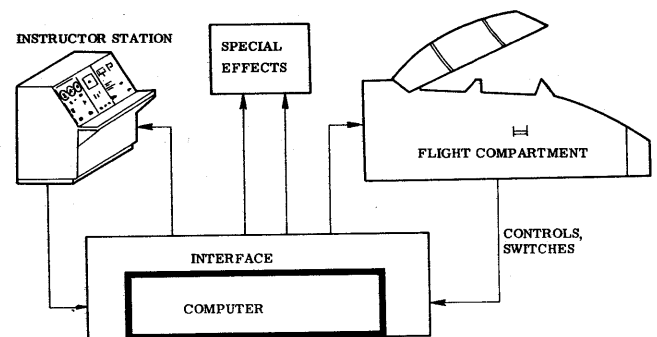
Clearly, the simulation of certain equipment, such as

aircraft, should incorporate physiological cues such as the trainee obtains in real performance through his senses. These are simulated to a degree by providing sounds appropriate to certain of the activities, appropriate force requirements to actuate controls, a motion system which jostles the man more or less appropriate to the maneuvers of his aircraft, and visual presentation through the medium of motion pictures or television.

The simulation which is carried on to a very high degree of fidelity is that required to "convert" pilot actions into realistic dial readings and other cues. Fidelity in anomolous operation, such as malfunctions, is emphasized in view of its importance in training pilots to make prompt and proper action under such circumstances.

Meanwhile, the simulator produces radio sounds for navigational purposes, drives the compass, etc. In places where noises appropriate to aircraft operation are likely to be encountered, the simulator activates the devices which produce these sounds. Examples of this are the brake squeal on landing or rumble as the aircraft is navigating across the airfield terrain. Simulators have several advantages over real aircraft in the training of pilots. First—and incidentally most compelling—is their reduced cost. In addition, certain technical features give them flexibility. Most noteworthy is the ability to introduce failures of various parts of the aircraft to train the pilot to react properly and safely under such circumstances. Thus, for example, engine failures may be simulated without danger. Similarly, tire blowouts, control system breakdowns to some degree, failure of part of the navigation system, may also be introduced to the

WHAT DOES THE COMPUTER DO?



The computer provides the model that represents the behavior of the aircraft systems.

Figure 2—General flight simulator

simulator to train the pilot. Not to be overlooked is the ability to simulate flight in controlled wind, controlled icing, etc., which is impossible to control when training in a real aircraft.

Modern simulators include other features which enhance the utility of the device in training. In particular, it is simple for the instructor to freeze simulation at a particular point where he intends to instruct the trainee in some phase of operation without completely losing the continuity. Other features such as being able to start the exercise at any point is valuable in situations where the trainee needs reinforcement in certain procedures at certain flight conditions. Recording of performance with subsequent playback permits the student to be a witness to his own activities, and consequently assists in his training procedure.

In the following sections we will mention some of these features again in terms of their impact on the programming of a flight simulator.

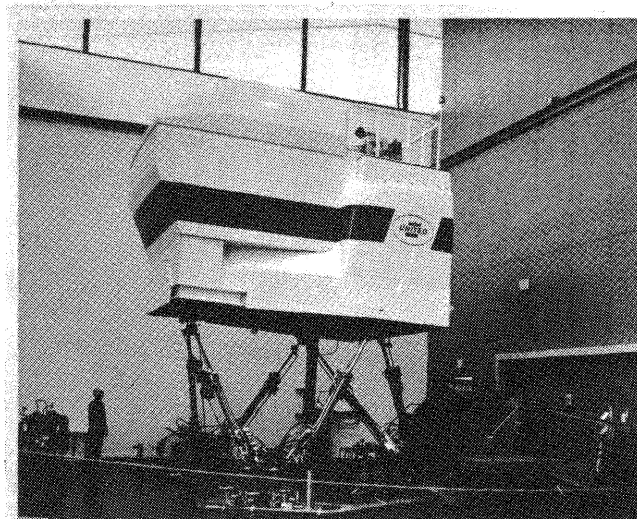


Figure 4—747 simulator with visual and motion

PROGRAM PREPARATION

Program preparation includes the programming and debugging of the individual programs prior to their integration in the simulator. In this context the programming process is essentially identical to that carried on in a laboratory computer. As seen, the main constituents are language translator, a debugging package, testing programs, data base management program, and the usual computer monitor features. In particular, the programmer will want to use the monitor's I/O and relocatability features as well as the set of utility programs during this phase of activity.

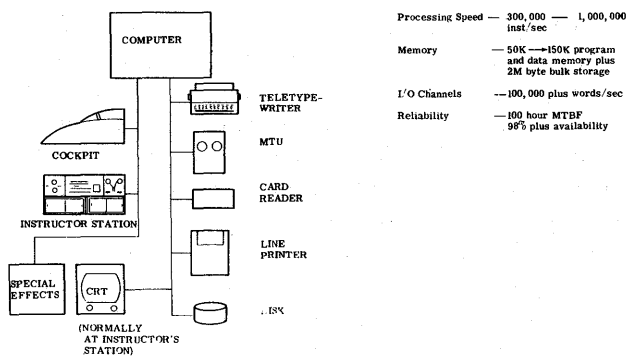


Figure 3—Overall computer requirements

In Figure 4 is shown a 747 simulator cockpit mounted on a motion system and having a VAMP* Visual System mounted on it.

Figure 5 is a representation of such a simulator with a cut-a-way showing the electronics used to compute the model. This is typical of modern jet simulator installations.

SIMULATION PROGRAMS

Figure 6 exhibits, in outline form, the programs which we shall discuss for the remainder of the talk. It is noted that the programs may be divided into two general categories—training and program preparation.



Figure 5—Typical 747 installation

* A trademark of Singer-General Precision, Inc.

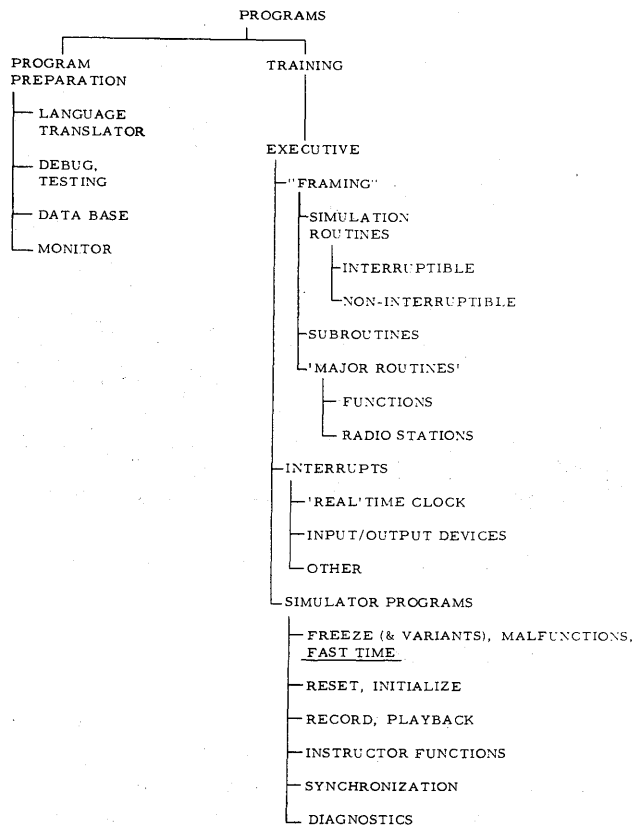


Figure 6—Simulator program tree

For many of our simulators we have used Assembly Language. This choice has been dictated by the very severe requirements of the real time environment in which we function. Recently, however, we specified and had written for us a compiler⁶ which produces fixed point programs with a degree of efficiency approximating that of a good assembly language programmer. Consequently, it is likely to be as good or better than an average programmer.

This language, FORTRAN-like in its syntax and semantics, is intended to be suitable for the documentation of the programs produced. Thus, it adheres very closely to normal mathematical rules. We imposed an additional constraint on the philosophy of describing this language in that we took a point of view contrary to that action by most higher order language developers, namely we restricted generality as much as possible. Our intent has been to provide this language with only those features which we could foresee as being necessary. This is an outgrowth of our anticipation that most programs written in this language would be written by people who are not programmers. Consequently, the tighter the language, the less likely they were to make subtle errors.

Debugging and testing is similar to normal functions with these names. The presence of a large data base and control of an executive is not unusual. A problem of integrating programs written by several people arises. Perhaps the problem of mathematical stability in the face of simplified routines is a bit more severe than normal. The problem of observing real time is to some extent postponed until debugging with the full set of simulator equipment. However, the trend is to force this problem earlier in the programming project.

Data base problems arise in that there are generally several tabular constraints imposed with sometimes contradictory requirements. Since duplicate storage requires extra memory in the delivered product, much effort is expended to avoid this "easy" solution.

The monitor and utility functions available with the purchased computer are exploited (sometimes after in-house modification) during test prior to testing with simulator hardware.

TRAINING PROGRAMS

In dealing with the training programs,⁷ we will find some to be unusual with respect to what is normally encountered in computer programming work. This is not to say that there is some magically different set of techniques, but rather that the intent and requirements of these differ from what we normally encounter in laboratory computer programming. The training program group is dependent heavily on a real-time executive. This executive is in many respects a scheduler as well as being a program which recognizes and reacts to interrupts and which guides the execution of simulator related programs as well as simulation related programs. The division of the programs for our purposes are the three major headings under "Executive," namely: "Framing," "Interrupts," "Simulator Programs."

Framing

The word "framing" in this context connotes the fact that the simulator programs are divided into programs which are operated at different iteration rates. Time is divided into intervals—frequently 50 millisecond in length—called "Frames." Since programs for different parts of the simulation operate at different rates, the programs which are executed in a given frame differ. The executive CALLS the proper programs in each frame. Simulation routines include all of the programs which provide the mathematical representation of the systems being simulated. They include all of the programs which compute the aerodynamics and engines, those which compute the various systems aboard, in-

cluding even such things as de-icer and air-conditioning systems. These systems must be simulated primarily to be sure that the pilot understands the indications of both correct and, perhaps more importantly, incorrect operation so that he is capable of taking prompt action when prompt action is required.

The programs dealing with applications like simulation of flight involve primarily solutions of the normal types of mathematical problems found in the field of mechanics. Consequently, most of these programs are heavily mathematical in orientation. A crucial issue in our solving these problems is that we keep the approximations as simple as possible in order to conserve both time and memory. We will always be guided by the recognition of the fact that the accuracy required for such solutions is frequently much less than it might be in an open loop engineering simulation.

One of the more widely discussed mathematical routines is that for integration. Frequently, expensive and fairly complicated routines are used for this purpose in mathematical computation. In our case we have the conflicting requirement that most of these more accurate routines require extensive computer time. Furthermore, since our step size is generally small, we find we can operate with a formula of the type of a modified Adams:

$$f_{n+1} = f_n + (h/2)(3f_{n+1}' - f_n')$$

In Figure 6 the words Interruptible and Non-Interruptible have been used as sub-headings under Simulation Routines. This choice of words emphasizes the fact that some of the programs involved in simulation are heavily time dependent in the sense that the time variable is an *explicit* independent variable of the function. Typically, these routines are integrations with respect to time. As a consequence of this dependence, these programs must be solved at fixed regular intervals of time—that is to say, in particular frames of simulation. The interruptible programs are those which do not exhibit this tight time dependency and as a consequence are flexible as to the frame in which they are solved, at least to a slight degree; i.e., can be completed in the next frame.

The subroutines as the next heading under Framing are simply the normal set of mathematical subroutines, including such routines as trigonometric and logarithmic functions. One of the problems encountered in the distant past was that of reentrant subroutines—arising from interrupts of some programs conceivably in the middle of the use of a subroutine. Many of our subroutines must be reentrant.

The major routines, while similar to similar routines used elsewhere, are of significant importance to the

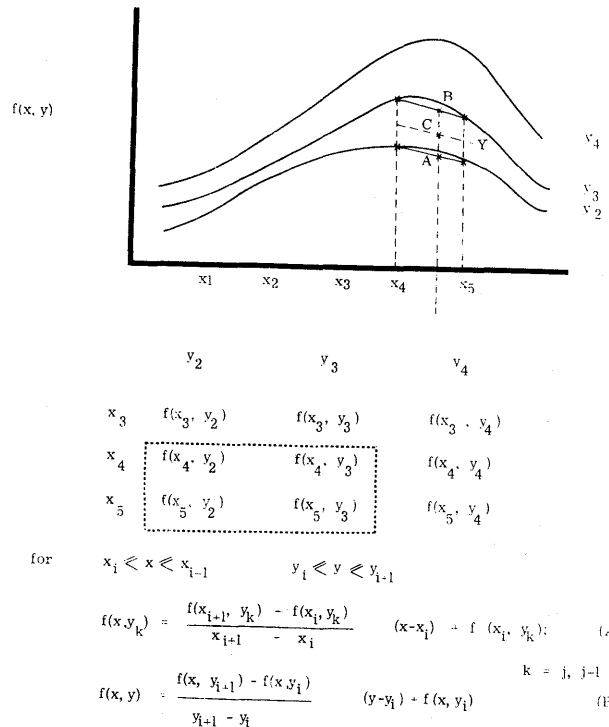


Figure 7—Interpolation

field of simulation to warrant their being identified separately.

The Function Generation routines are used for computation of over 300 functions of one and two variables in a large four-engine jet transport. Thus, much core is required as well as a noticeable fraction of the available computer time in a general purpose computer.

Interpolation, as we implement it, is represented in Figure 7. At the top is shown a function of two variables, $f(x, y)$. This function is represented in memory as a table of values, where the arguments may be explicit (in some manner similar to that shown) or normalized and implicit. The small x 's plotted on the curves represent the values entered in the table.

Normally, interpolation proceeds as two interpolations "on x " as in Equation A to obtain, for instance, points A and B. These are followed by an interpolation "on y " to obtain point C, using form B. Examination shows this method to represent the first four terms of the Taylor Series expansion of a function of two variables. The method of interpolation has been selected due to the ease of changing data points 'at the last minute' when required (and it is frequently required).

Polynomial approximations have the characteristic that a change of a single data point affects all coefficients. This frequently proves to be an awkward constraint.

The process of interpolation as described above involves a search for the "adjacent" tabular values, given an argument (x, y) . This search, as well as the calculation, is minimized in order to minimize the computing time used in the process.

The problem of simulating radio navigation facilities in a flight simulator is in some sense similar to the previously mentioned problem of "function generation." In another sense, the problem is different.

In simulating the radio facilities, it must be observed that the trainee has at his disposal all of the normal normal types of radio navigation equipment normally available in the aircraft being simulated. This includes such things as UHF, VHF, omni-range, and Instrument Landing Systems (ILS). The trainee may tune any of these sets at any time that he pleases. It may be parenthetically remarked that he does not do these things capriciously or at random. However, it is difficult, if not impossible, to take advantage of any prediction in terms of his use of this equipment to reduce program requirements except for the slight advantage which we may gain by noting that he normally cannot be tuning all of the radio facilities simultaneously.

In selecting a radio station the objective is to determine whether the receiver is tuned close enough to the frequency radiated by the transmitter such that were it within range it would be within the band pass of the receiver. In tuning a radio station the program must locate any station which satisfies simultaneous constraints on frequency and range. Furthermore, the implication of the transmitter being within range is clear. Thus, the initial problem is one of table look-up in which a determination is made on the following two points:

- a. Is the frequency to which the receiver is tuned close enough to the transmitter frequency such that the transmitter signal can be passed through the receiver?
- b. Is the location of the aircraft close enough to the location of the transmitter for the given transmitter power such that a signal of sufficient strength would have been received by the real aircraft radio equipment were the real receiver tuned to that station at the same range? This is typically simplified to an x and a y calculation rather than a range calculation.

Several factors influence the approach taken in programming this activity. Since the various types of sta-

tions are generally received on separate receivers, it is clear that the first step in proceeding is to detect the type of receiver being tuned and search only the table containing simulated transmitters acceptable to that receiver. A second factor is the necessity for rapid response. It is required that, as the trainee tunes the receiver, he hears the little "blips" that he would normally hear as he tuned through stations. If the trainee is tuning from, for instance, channel 12 through channel 32, and channels 20 and 24 are within range, as he passes through these during his tuning, he should hear a response from the radio.

There are occasionally duplications so that two radio stations having the same frequency but at different ranges may exist. It is necessary, if the primary search is on frequency, to have a secondary search on range to eliminate stations which are within prescribed range. While it is possible to arrange the stations within each category to be monotonic, it is not generally true that the frequencies will be equally spaced. In fact, as mentioned before, there may be repetitions of some frequencies. A search algorithm⁸ which has produced excellent results for us has been developed. This algorithm may be characterized as being "real-time" in nature. It remembers where it was the last time it was used and consequently in view of the continuous nature of most tuning it reduces the amount of search to find the station which is being sought.

After the radio station has been identified the variety of information germane to this radio station is located in a block, "unpacked" and presented in another block to the computation programs related to radio navigation.

Interrupts

The simulator computer programs are dependent in significant ways upon interrupts. An interrupt is a signal, generally from an external source, which causes a discontinuance of the normal sequence of calculation accompanied by storage of sufficient information to resume this calculation upon command, and by a transfer in control (jump) to a prescribed location in memory corresponding to the interrupt which has occurred.

One of the primary uses of interrupts is the insertion of a relatively precise time signal at stipulated intervals. This signal occurs, in most cases, every fifty milliseconds. This signal is used by the programmers as representative of real-time input. In a sense it is the means of synchronizing the operation of the simulator computer with the external world.

As previously noted when discussing "framing" the computer executive must be aware of real-time, or at least that which both the computer and its immediate environment believe to be real time. In this context, of course, the trainee is construed to be part of the "immediate environment." Using these time signals the computer may conveniently count the frames. In most cases sixteen frames are grouped together as a "Cycle." The choice of sixteen is more or less obvious in that the rate of solution of programs requiring different rates may be conveniently halved as the time requirements become less stringent permitting then rates with ratios of

16:8:4:2:1

This framing has other implications. The most important of these is perhaps the situation which arises when two computers are operated together in a large simulator. Under these conditions, it is necessary that the two computers not only be starting each frame together, but, perhaps more importantly, that they start each Cycle together. This synchronization requirement is a direct result of the requirement that information transferred in either direction between computers be properly phased (if I may use that word) in order to observe stability requirements in the solution of differential equations.

Input/output interrupts are fairly standard. In our applications we have used multiplexed channels for the obvious reasons. The channels have been allocated to both the normal computer type peripheral equipment such as card readers, line printers, as well as the real time equipment such as D/A converters and digital word channels.

On some simulators involving multiple processors and on some earlier simulators using somewhat slower computer equipment with somewhat different I/O facilities, several additional interrupts were required. In one case an interrupt was required between computers to indicate the fact that a transfer of data between the computers in a given direction had been completed. This was used by the computers to set up for transfer in the other direction. In fact, in an earlier simulator using three separate computers without any shared memory, there were actually six data transfers to be made. This inter-computer interrupt was essential in this situation. It was in this same project where a separate I/O control box was developed and this provided with its own interrupt. In this case a computer not involved in inter-computer transfers was being serviced by the external real time input/output equipment and required that interrupt for proper operation.

Simulator programs

Specific to training itself as a general problem are some programs, which we have referred to as simulator programs, used in the educational process. These programs are not involved in the simulation of any of the equipment used by the trainee. Nor are they involved with the operation of the computer as a device containing a model of the thing simulated. In general these programs are programs which provide functions which could either not be done in the live equipment or could be done only at unnecessarily high risk (in an airplane).

Malfunctions

The first of these is called "malfunctions." The term, I believe, is reasonably explicit. The intent of these programs is to provide a simulation of some system of the aircraft when operating incorrectly and to exhibit appropriate indications of this faulty operation. Typically, in the programming of a simulator, this might be represented as a coefficient on an equation representing some performance. These failures show up in several ways. Some failures are complete; thus, an output might be either a "normal" value or a constant (usually zero), depending upon whether the system being simulated is intended to be operating or "failed." In other cases the failure might be partial. In such cases the math model would have a coefficient which might be varied from unity (for normal operation) to some other number to indicate the failure. The control of these malfunctions is generally at the discretion of the instructor. He has means of introducing into the computer a number representing the malfunction and a switch indicating that the malfunction shall become effective. In terms of a programming of such a situation the equation to be solved representing the system is normally preceded by a check of the malfunction. Should the malfunction be total, a complete branch around the calculation may be feasible with the insertion of a constant as the pseudo result for the malfunctioned system.

The instructor who has carefully planned his training session will have a script indicating which malfunctions shall be introduced and under what conditions. Clearly, this indicates that the control of the introduction of malfunctions could be a computed operation. In fact, this is sometimes done. The price of doing this is not only the price of the additional program to continually check these conditions (time or other conditions), but the additional program which permits the instructor to

enter the malfunctions that he wishes and the conditions under which they are to occur. These data will change from training session to training session. But the price is not completely paid yet. On top of this, it is almost always required that the instructor have the prerogative of overriding even those malfunctions that he has scheduled. Thus, additional programs must be introduced to check his "override" control so that if he chooses to eliminate the malfunction for some reason such as the training mission not going as planned, he can do so.

Freeze

Freezing or halting of the simulation is another feature almost always included in a training simulator. The instructor can, upon command, stop all computation. He does this in some cases in order to converse with his student and instruct him on the correct action or procedure. This feature is used normally only when there is a gross mistake which the instructor feels is so important that it needed to be corrected before the student is influenced by any repetition of this mistake. Typically, this feature is accomplished in the program in an indirect way. By this I mean the typical operation is to set any variables representing increments of time to zero. Thus, time stands still. On the other hand, it is generally of supreme importance that the output be continued in order to keep all of the displays at their readings just prior to freeze. It may be essential to the instructor's purposes to refer to these readings in his explanation to the student as to what should have prompted him to do something other than what he has done. Inputs are also typically permitted during Freeze—not so much for the function during Freeze only, but because of their necessity during Reset or Initialization.

Parameter freeze

Another form of "freeze" is the so-called parameter freeze. This term refers to the fixing of a given variable in the flight simulator. For example, it may be desirable to relieve a pilot of the necessity of controlling a variable in order to reinforce his learning the control of some other variable. To this end, parameter freeze is used. Thus, for example, the altitude may be maintained at some fixed number if the primary intent is to train the pilot to use radio equipment for navigation. If, for instance, it is desired to emphasize the training of keeping an aircraft on the localizer, the required rate of descent might be frozen so that the pilot is relieved of the necessity of keeping the aircraft on the glide

slope. The implications of this on the programming are, of course, clear. The computation of the number may be inhibited, or more likely, it is allowed to continue, but the program simply inhibits the storage of the answer computed in preferring to insert a constant in its place.

Fast time

In some simulators, notably space simulators, 'fast time' has been a feature of the simulator. In this situation, the training is suspended temporarily and the time increment is increased by a factor simulating passage of real time faster than the actual time elapsed. This may be useful in some circumstances to exhibit to the trainee the long term effects of an action he has taken. This feature is intended to show the trainee the effects of severe errors. Typically, fast time has an attendant inhibition of any action by the student affecting the simulation.

Reset, initialize

Reset and initialize are two words for the function which permits the simulation to be established at an a priori condition for training. Thus, one condition might be the aircraft at the beginning of the runway prepared to take off. Another condition might be the aircraft in full flight at a given velocity, altitude and attitude preparing to make an approach to landing. These conditions are stored in bulk memory of some type, usually magnetic tape. They could, however, be introduced through a medium of punched cards or punched paper tape if a severely reduced amount of equipment is necessary for economic reasons. It is to be remarked that when the flight simulator is being reset or initialized, it is in a freeze condition. Thus, the input/output equipment is working so that when the new conditions are in memory the meters, lights and other indicators are all driven to the proper condition while the aircraft is still in freeze. The controls are set manually and accepted as input. This is necessary in order that the trainee may adjust himself to these values prior to his having to actually "fly" the airplane.

Record-playback

A relatively recent requirement introduced into the simulator business has been the "Record-Playback" feature. The objective of this operation is to record significant variables during the course of training exer-

cise. After the exercise has been completed, the recording is played back driving the required instruments to review for the trainee what his performance was during the training exercise. Properly used, this feature is a great training aid in that the instructor, knowing what is coming up, can indicate to the trainee the effects which are about to occur in response to the trainee's action. The major problem in dealing with this feature from a programming standpoint is the control of the number of variables to be recorded. The programmer looks for those variables which might be construed as, in some sense, basic. This may be interpreted as the smallest set of variables which, with the aid of normal simulation computations, can cause the proper instrument readings to be reproduced reflecting the pilot's performance.

Instructor functions

I refer to the various controls which may be exercised by the instructor at his console. Some of these have been mentioned previously in our discussion of the instructor's control of Malfunctions. Implicitly he also has controls for Freeze, Parameter Freeze, Reset and Initialize, Record and Playback. Programs to examine the inputs from these various instructor stations are clearly required.

In addition, certain other conditions may be required for simulation and ordered by the instructor. Winds and turbulence in varying amounts may be introduced by the instructor for training of the pilot under these conditions. Icing on the aircraft or snow, slush or ice on runways may also be introduced for the obvious training purposes. These inputs and their attendant effects require additional programs to be introduced. Typically the outputs of these programs are used to modify some variable of the computation. The method of programming is similar to that used in the introduction of malfunctions (malfunction of the weather?).

Synchronization

In those simulators using more than one central processing unit, additional programs dealing strictly with computer operation, specifically and most importantly synchronization, are also required. Although the real time interrupt can be relied upon to keep the two simulator computers in synchronization once started that way, it is necessary to insure that they start simultaneously. Once this has been arranged, it is a simple matter to make the program check from cycle to cycle that they have remained in synchronization.

Thus, any transient failure of one of the real time interrupts may be overcome in some cases.

Diagnostics

Diagnostics are, of course, required. In addition to the normal mainframe and central processor unit peripheral diagnostics, additional diagnostics for the real time input/output equipment and for the cockpit equipment are generally required. These diagnostics are made automatic to the greatest extent possible. However, much of the cockpit equipment involves human interaction in either reading or observing the results of a computer output or in controlling something which results in a computer input. In view of the size of the complete system, it is clear that a reasonable set of diagnostics is mandatory. The problem involved in reaching this "reasonable" set is important since the objective of keeping the equipment on the air and the objective of reducing the length of time it takes to run the diagnostics conflict. Delicate compromise is required.

SUMMARY

Programming of training simulators involves a combination of scientific, real-time and multiprogramming problems. During program development the activity is similar to laboratory scientific programming. Throughout the process emphasis is on efficient core and time utilization characteristic of all real-time programs. The final program being a combination of many programs operating under a real time executive resembles a multiprogram operation. Much of the program is devoted to simulator as contrasted with simulation related programs.

REFERENCES

- 1 *Math model compiler reference/operating manual*
Internal Technical Report Singer-Link Division
- 2 B M TATE
Boeing 747 training developments and implementation
Fourth International Simulation and Training Conference
Society of Automotive Engineers May 1971
- 3 J A FERRARESE
Assessment of new training systems as substitutes for airborne training
Fourth International Simulation and Training Conference
Society of Automotive Engineers May 1971
- 4 R L TAYLOR A GERBER

- A study to determine requirements for undergraduate pilot training research simulation system (UPTRSS)*
Air Force Human Resources Laboratory AFHRL
TR 68-11 1968
- 5 E COHEN
How much motion is really needed in flight simulators
Fourth International Simulation and Training Conference
Society of Automotive Engineers May 1971
- 6 *Math model compiler reference/operating manual* op cit
7 R E FLEXMAN W P JAMIESON
J M WALSH et al
Synthetic flight training system (SFTS) concept formulation report
Technical Report NAVTRADEVCEEN 68-C-0106-1
July 1968
- 8 Internal Report Link Division

The handling qualities simulation program for the augmentor wing jet STOL research aircraft

by WILLIAM B. CLEVELAND

NASA-Ames Research Center
Moffett Field, California

INTRODUCTION

Aircraft have been simulated on computers for a variety of reasons. The training of pilots and crews on operational flight trainers, for example, is a common use of simulation. Subsystems of aircraft are often simulated to firm up the design of the hardware and frequently the whole aircraft must be simulated to help in the design of the subsystems. This is the case in the simulation of the Augmentor Wing Jet STOL Research Aircraft, a modified de Havilland C8-A Buffalo, in which the total aircraft was simulated to determine final design values for control systems and devices which augment control of the aircraft. For research and development simulations such as this one, simulation software and hardware must have general application while in piloted "man-in-the-loop" simulations speed of computation is the overriding concern. Thus the aircraft model and computer software and hardware must be merged to provide an accurate simulation which meets the needs of the research objectives.

The STOL problem

Short Take-Off and Landing (STOL) aircraft are designed to use 1500 ft. runways as opposed to 10,000 ft. runways commonly used by commercial jet transport aircraft. To meet this requirement it is necessary to fly at a slower speed with a resulting steeper flight path angle. Due to the slow speed requirement all aerodynamic control of the aircraft is reduced as aerodynamic control power is proportional to the square of velocity. For example:

$$\text{Lift} \propto V^2 C_L$$

The coefficient of lift C_L is a function of the shape of fixed parts such as the fuselage and wings but it is varied by movable surfaces such as flaps and spoilers.

As the aircraft velocity decreases or increases C_L must be increased or decreased to maintain the required value of aerodynamic lift. When conventional means cannot produce sufficient lift special devices are a necessity. For example, if insufficient lift is obtained from aerodynamic properties, direct thrust lift from the jet engines may be employed. Similarly, lateral-directional (roll-yaw) control is reduced in the same manner as lift at the lower speeds making the STOL class of aircraft in general dependent on special aids in roll-yaw control as well as lift. For the C8-A a special aerodynamically high lift flap is used in conjunction with vectorable jet engine thrust to provide the necessary lift at low landing approach speeds.

Handling qualities

Along with loss in control effectiveness the stability of flight maneuvers is also reduced and improvements in stability as well as control must be introduced to bring the aircraft "handling qualities" up to an acceptable level. "Handling qualities" is a general term in which the aircraft characteristics are rated by pilots on a scale ranging anywhere from "uncontrollable" to "optimum." The pilot must be asked to perform a specific task. For instance, he may rate the handling quality of an aircraft in a normal landing approach as "optimum." However, the same approach may be difficult or nearly impossible with an engine failed and his rating would be consequently lower. The ratings are nearly purely subjective, as the rating is the pilot's opinion. Normally several pilots are used to avoid personal biases and obtain a crude statistical sampling for a handling quality rating. Since one of the primary outputs of a simulation of this type is the subjective pilot evaluation, systematic investigations require exacting simulation models and the ability to introduce or repeat any combination of initial conditions, failure

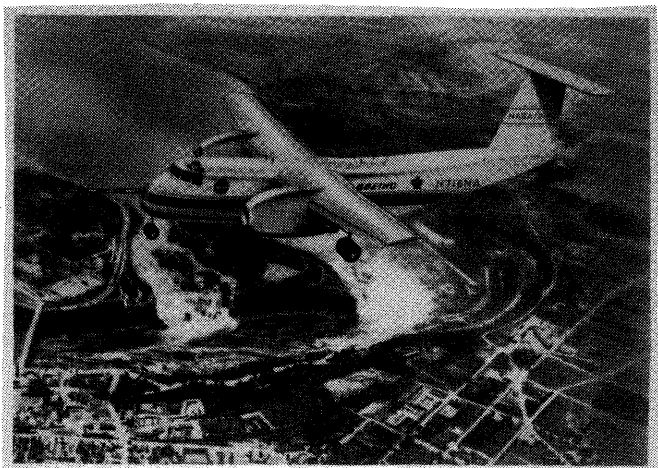


Figure 1—The augmentor wing jet STOL research aircraft

modes, control effectiveness, air roughness, etc., desired to obtain reliable pilot ratings.

STATEMENT OF PROBLEM

The augmentor wing aircraft

The Augmentor Wing Jet STOL Research Aircraft is sponsored jointly by the National Aeronautics and Space Administration and the Department of Industry, Trade and Commerce of Canada. Major modifications of a de Havilland C8-A Buffalo are presently under way to provide the research vehicle, Figure 1.

To meet the short field requirements set by the Federal Aviation Agency for STOL aircraft the airplane contains several novel pieces of hardware. The augmented jet flap is a high lift device which gets its name from the blowing of a flat jet of air down the slotted flap as seen in the wing section diagram, Figure 2.

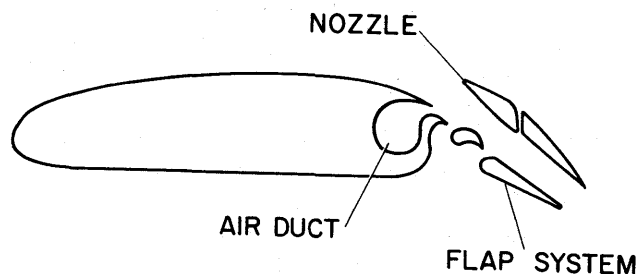


Figure 2—Cross section of the augmented jet flap wing

Cold air from the fan-jet engines is ducted to ejector nozzles to provide the air jet. The ailerons also contribute lift by drooping in conjunction with the flaps but only up to one half the total flap deflection. The aileron is a boundary layer control device in which air is blown over the surface to improve the aerodynamic force characteristics. Normal roll control by the ailerons is provided by differential aileron deflections about the common flap-aileron angle operating point.

The effect of air blowing on the aileron may be seen in Figure 3, the function of coefficient of roll $C_{l_{\delta a}}$ versus aileron deflection δa and blowing coefficient, C_{J_a} . The coefficient C_{J_a} is a measure of the cold air thrust, T_c , non-dimensionalized through division by the product of dynamic pressure and wing area, ($C_{J_a} = T_c / \bar{q} S$).

As C_{J_a} increases for any down going aileron angle ($\delta a > 0$) so does the rolling moment on the aircraft. It is apparent that boundary layer control adds great effectiveness over the non-blown aileron. Since the roll coefficient curves represent an individual aileron the total rolling coefficient for both ailerons is $C_{l_{\delta a}}(\text{port}) - C_{l_{\delta a}}(\text{starboard})$.

The engines themselves are remarkable in that they provide the air for the flap blowing and aileron boundary layer control and more so since the engine thrust is deflectable. The thrust angle, under pilot control, is normally directed aft for cruising but it may be directed straight down to provide direct engine lift at slow speeds.

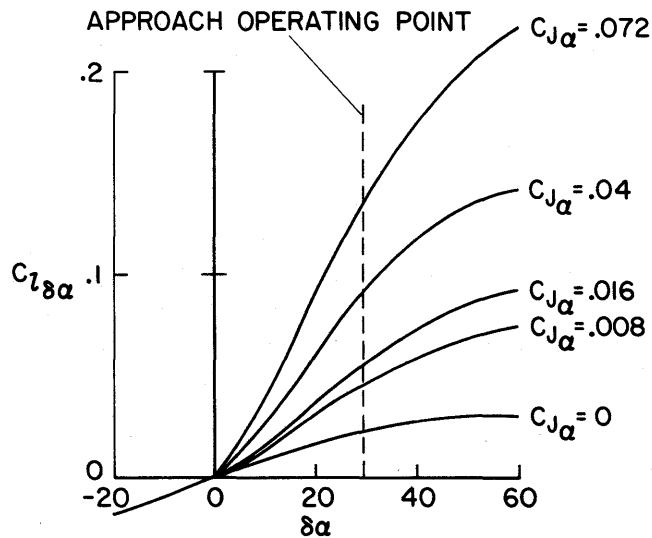


Figure 3—Rolling coefficient $C_{l_{\delta a}}$, a function of aileron deflection, δa and blowing coefficient C_{J_a} for one aileron, down-going is positive

Objectives of the program

An early piloted simulation of the modified Buffalo was conducted to determine just how the pilot might best control the aircraft in takeoff, transition to cruise and back to landing, and landing itself. The results showed that the aircraft overall had acceptable handling qualities, but the pilot's workload was higher than desirable for a commercial STOL aircraft. However, it was felt that the use of Stability Augmentation Systems (SAS) in lateral (roll) and directional (yaw) would reduce the work load to a satisfactory level. Systems that were designed to help the lateral characteristics provided turn coordination and dutch roll damping. In order to evaluate these control aids the aircraft was simulated on a moving base simulator in as much detail as possible. The test pilots had not only the normal flight instrumentation but good visual and motion cues to make their evaluation of the proposed stability systems. Various effects such as control system hydraulic failures, SAS servo failures, and engine failures were needed in addition to the usual disturbances such as air turbulence to evaluate the handling qualities of the aircraft in both normal operating and failure modes.

Computer requirements of the handling qualities simulation

The problems discussed in these previous sections place certain requirements on the simulation and at the same time allow some few concessions. To be specific the simulation of this vehicle must provide the following:

1. Six-degree-of-freedom equations of motion completely rigorous and no approximations; a flat earth is acceptable in a landing study.
2. Accurate and detailed aerodynamic derivatives including all coupling effects. This is required to access handling qualities.
3. Engine performance model.
4. Stability augmentation system to help make the aircraft easier to fly in the lateral-directional modes.
5. Air turbulence model, wind shear, gust upset, and/or steady state winds.
6. System failures, engine failures, SAS failures of several types.
7. Non-steady aerodynamic effects; for example, the effects of a time delay from when the air flows over the wing until it reaches the horizontal stabilizer.
8. Landing gear model for landing and roll out.

Other simulations, especially of high speed and very large aircraft, would require a representation of body bending modes and aero-elastic effects.

The amount of computation required to accomplish these items listed above made it impractical, if not near impossible, to do with analog computers (at least with the desired accuracy). Thus this simulation was done using a digital computer.

SIMULATION HARDWARE AND SOFTWARE

Simulation computing system

The Ames Research Center simulation computing system is made up of both analog and digital computers. The principal component of the system is the EAI 8400 Digital Simulation Computer. This computer is a 32 bit, 32,000 word machine with a real-time interval timer and floating point hardware. Its input/output to the analog domain consists of 32 bits "in" and 32 bits "out" of discrete on-off signals as well as 64 channels of multiplexed analog-to-digital converters (ADC) and 64 DACs. Peripheral equipment includes four magnetic tapes, line printer, card reader, disk file, and typewriter. Of this equipment 29,500 cells, 64 DACs, 16 ADCs, 17 discrete bits "out" and 12 bits "in" were used in addition to the computer peripherals. The memory was allocated as follows: 6,500 for the simulation monitor system, 21,000 for the simulation program, and 2,000 for the simulation software (user provided). Because most of the cockpit instrumentation was developed in the past for all-analog simulations, data communication with the simulator cab is entirely through the ADC-DAC linkage and the discrettes. No digital instruments were used.

In addition to the digital computer an analog computer (EAI 231-R) was used. Whereas the digital computer computes the aircraft model as its primary work, the analog computer is used as a buffering device between the digital computer and the analog recorders, motion and visual simulators, and the various cockpit instruments and controls.

Theoretically the analog computer was not needed since no part of the aircraft was simulated on it but with the myriad of devices requiring data transfer to and from the digital computer it is impractical not to have some sort of analog computer for a data trunking center. No claim is made to call this a hybrid computing arrangement due to the nature of the workloads on the two systems, however, it is interesting that the requirement exists for analog components to be available for the "digital" simulation.

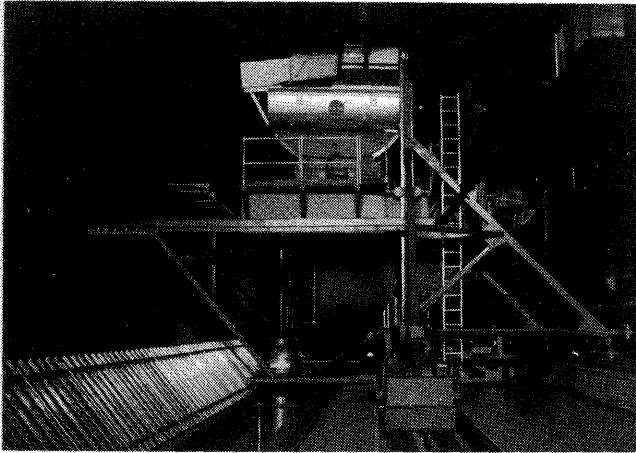


Figure 4—Flight simulator for advanced aircraft located at NASA-Ames Research Center, Moffett Field, California

Simulator system

To obtain the most valid handling quality evaluations the pilot must be subjected to as many motion, visual and aural cues as he would experience in flight. While this is impossible to achieve on a ground based simulator the most important cues of STOL aircraft can be faithfully duplicated on the large motion generator at Ames called the Flight Simulator for Advanced Aircraft (FSAA), Figure 4. This motion simulator has a fully instrumented cockpit and six-degrees-of-freedom travel capabilities of ± 50 feet in lateral, ± 4 feet in vertical and longitudinal, and at least ± 22.5 degrees in roll, pitch, and yaw. The FSAA was chosen for its large lateral travel which proved most useful in the roll-yaw control handling qualities study and also in the engine-out maneuvers. In the simulator the pilot has the capability of flying on instruments or by visual contact. The visual scene is an out-the-window pictorial representation of a landing field and surrounding countryside. The scene is a television representation in which the model of the landing field is scanned by a color television camera mounted in gimbals so that in addition to three translations the angular motions of roll, pitch, and yaw of the airplane are displayed to the pilot.

Ames simulation software

The simulation software system at Ames has evolved from manned simulation requirements. With a "man-in-the-loop" all work must be performed in real time. Historically, manned simulations have been done on analog computers with their fast parallel computing

capability and only recently have digital computers been used widely for the real-time problem. Execution time is of prime importance, thus the software used in simulations must meet rigorous execution time requirements. At present the Ames simulation software is of two types: hardware support and program support. The program support software is a group of programs under the label FAMILY I.¹ The main features of FAMILY I include the basic Ames simulation monitor, integration packages, real-time magnetic tape data dump and two special systems—MOTHER (Monitor Time Handling Executive Routine) and CASPRE (Comprehensive Aid to Simulation Programmers and Engineers). The hardware support software serves to make analog type operations tractable from the digital computer. This software is critically important to the operational efficiency of Ames' simulations.

Program support software

A real time scheduler called MOTHER was developed in response to the problem arising from the speed limitations of the EAI 8400 digital computer. It was recognized that programs that contained high frequency systems or that sampled high frequency analog inputs must sample and solve the system equations at high rates to produce sufficient accuracy. However, the size of our simulations indicated that one big program loop solving all the systems would normally produce integration and sampling step size too large to accurately reproduce the high frequency portion of the problem. The easiest solution to this problem would be to get a computer fast enough, but when the problem is not too large and the frequencies are not too high, there are less drastic solutions. The approach used in the Ames' simulations is to do the high frequency operations more often than the low frequency ones using a special piece of software called MOTHER.

MOTHER is a scheduling and executive routine which schedules subroutines to operate at synchronized time intervals or to operate within given time constraints. Subroutines are defined to MOTHER to operate within specific time constraints. By organizing all the high frequency computations into one list and the remainder into a second list one may define to MOTHER what the computation rate is to be on each list. Both lists are called constrained processes because they must be completed within time constraints as opposed to synchronized processes such as input/output of analog data which must be performed at specific time intervals.

In Figure 5 a two loop program is shown as it might be run under a MOTHER schedule. Assume that the I/O for loops A and B must be executed every 10 and 20 milliseconds respectively and that the computation blocks, A and B, must execute within the same 10 and 20 milliseconds.

Since the I/O processes are synchronized, they are executed first. After the I/O the constrained processes begin; process A has been scheduled to execute first since it must execute within the shorter time constraint. At the completion of A, process B begins; at the time of 10 milliseconds the I/O of A interrupts the constrained process and proceeds to execute. Note that A executes again before process B resumes its execution and finally is completed. At this point all the synchronized and constrained processes are completed for this period. The period as used in this context is the shortest time into which all the various process times must divide integrally. The ability of MOTHER to do this scheduling is a great aid to the simulation programmer since all he must do is to provide the simulation subroutines and specify the calculation rates. The scheduling burden has been removed.

Other items that have proved useful are MOTHER's executive features such as the servicing of simulation mode control and discrete signal inputs. In the typical operating environment a request for a mode change may come from any of three areas: The analog domain, the program itself, or from the digital control console. MOTHER receives the mode requests and services them by first setting a user-provided mode word and then executing the process list defined for this mode. Discrete bit signals from the analog domain are serviced

by either setting a corresponding Fortran word high or low according to the condition of the input bit or by executing special subroutines not in the lists of defined constrained processes. Figure 5 shows the mode and discrete bit servicing periods which follow execution of the defined processes.

The digital simulation computers are completely dedicated to simulations as Ames' batch work is performed at a separate central facility. Hence the emphasis in the simulation laboratory is not on computer throughput but on computer/simulator uptime. In this context computer uptime means not only that the hardware be operational but also that the program be operationally useful. Experienced simulation engineers will certainly agree that changes to programs occur at a high rate and in a seemingly never ending stream. In this state of flux a means of quickly updating and changing data and equations is a necessity. Since the simulation programs are written in Fortran, recompilation is the only practical means of making lengthy changes. However, small changes may not warrant delaying an operational simulation to make a time consuming compilation. The software used to make minor changes is called CASPRE.

Changes of data constants, for example, are very simple, e.g., if it was desired to set the weight of the aircraft to 100,000 pounds the computer operator need only type `+WEIGHT=100000$`. The elements of this statement fall into several categories. The `+`, `=`, and `$` are commands or operatives defined to perform specific tasks on the character strings `WEIGHT` and `100000`. The plus sign indicated to CASPRE that a Fortran name was to follow, the equal sign is a command to set the "weight" cell to the following data concluded by the dollar sign. This flexibility and ease of change is essential to the operation in research and development simulations.

Some of the many directive codes in the CASPRE system include typewriter display and modification of a data cell's contents in floating point, octal, or integer formats and even in Binary Coded Input (BCI) if the cell happens to contain such data. The display is also available on the line printer making it possible to provide short data print-outs. In practice this print capability is rarely used for more than program check-out.

Small program changes are made with a combination of machine language instructions and symbolic addressing. Making these changes requires some knowledge of the machine language codes but the nature of program patches normally requires only arithmetic plus a few conditional branch instructions. For example, a very common request is for a sign change in an equation. This is easily done by a simple operation

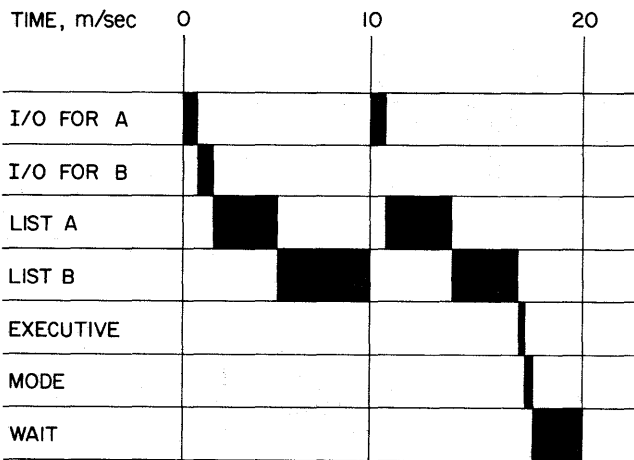


Figure 5—MOTHER schedule for two computation loops

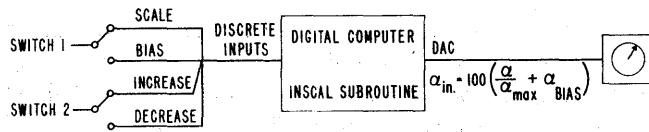


Figure 6—Calibration of cockpit instruments using INSCAL

such as changing an add to a subtract instruction. Large changes such as the insertion of an equation into the program are made by a jump from the compiled program to an area of memory set aside as a patching area, where the equation is patched in machine language and then a jump back to the equation stream.

Hardware support software

One of the operational problems associated with flight simulators is the calibration of the cockpit instruments. The instrument readings are, for the most part, linear with voltage input but the zeroes and scaling vary with each instrument. It is a daily process to calibrate and check each instrument used. Calibration consists of both a bias and a gain on the problem variable. It was the practice at Ames to provide a DAC channel scaled in some convenient manner and with an analog computer do the necessary voltage scaling and biasing before sending the signal to the instrument. This straightforward approach proved to have a major fault albeit a human one. Due to the general purpose nature of the computers and the simulators there is a large turnover of programs on the equipment. The analog equipment requires a high amount of human attention to keep up with changes which frequently resulted in improperly scaled instruments and control inputs thereby plaguing the uptime record of the facilities. The solution to this problem was to scale and bias the instrument drive signals in the digital program before sending them out via the DACs. INSCAL was devised to ascertain just what values of gain and bias were required. Figure 6 shows two switches and an instrument located in the simulator cabin. A computer operator selects the instrument to be scaled by typing the DAC channel number into the INSCAL routine. From this point one of the simulator personnel in the cockpit performs bias and scaling. When the operator sets switch 1 to Bias, the variable to that DAC is set to zero and the DAC output is only a bias voltage. If the instrument is not in its null position, switch 2 is used to increase or decrease the value of the

bias until it does null. Having determined the voltage bias of this instrument, switch 1 is set to Scale position. Internal to the INSCAL program a static test value has been previously assigned to the variable to aid in the determination of its scale value for the instrument. With switch 1 in the Scale position the variable is set to its static value. The operator again uses switch 2 to increase or decrease the scale factor until the meter reads the prescribed test value.

For most simulations this process is repeated for 10 to 30 instruments. However, once the initial cockpit calibration has been done the gains and biases, having been saved in the program, are available for subsequent setup of the program. Normally, subsequent calibration checks require no changes to the stored gain and bias values, thus dramatically reducing the setup and turn-around time for simulations.

A requirement for many channels of recorded data on analog strip chart recorders prompted the multiplexing of 2 variables onto one recorder channel. Multiplexing can be done by mechanically switching between contacts carrying the appropriate signals. This approach will require many DACs in the digital simulation. By multiplexing the variables in the digital computer the same effect is produced, but only one DAC is used to display two variables. Figure 7 shows an example of what can be done with multiplexing. The input and output signals of a control system have been multiplexed onto one data channel for comparison. Normally high frequency and/or discontinuous signals are not multiplexed since in that form they would be very hard to read. Most variables can be multiplexed in aircraft simulations and the use of this routine has proved to be quite successful.

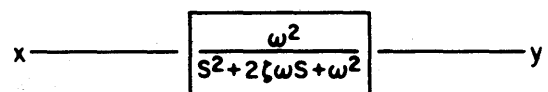
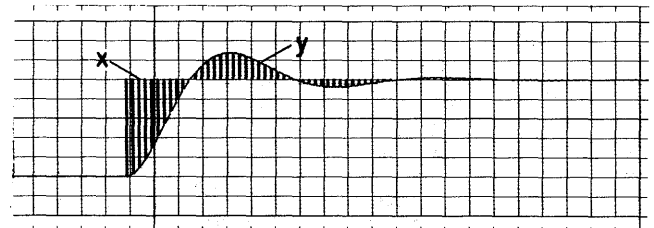


Figure 7—Strip chart recording of X and Y multiplexed for comparison

AWFTV PROGRAM MECHANIZATION

The organization of the digital program is illustrated in Figure 8. The middle block contains MOTHER, the executive program for the aircraft and support sub-routines. The upper block in the figure represents the main program of the simulation which defines to MOTHER the timing requirements for the execution of programs and input/output data transfer schedules.

The three lower blocks constitute the simulation which consists of two real-time calculation loops and a set of supporting subroutines. Because MOTHER can schedule the computations of fast variables more frequently and slow variables less frequently, MOTHER makes it possible to run complex programs with adequate dynamic fidelity than otherwise is possible using a straightforward serial calculation of all variables. Two calculation loops proved satisfactory in this simulation. The high frequency loop contained the rotational kinematics, part of the rotational aerodynamics, the control system and the landing gear model. The low frequency loop contained the translational kinematics, the remainder of the aerodynamics, the engine model and the simulator drive calculations. The terms "high" and "low" frequencies are relative, of course, and the split of the workload is somewhat subjective. Usually the rotational behavior of an aircraft in flight contains higher frequencies than the translational. As a natural grouping of rotationally oriented work the control system and rotational aerodynamics were put into the fast loop with the rotational kinematics. The landing gear equations must be solved relatively fast due to the high transient frequencies present upon touchdown. The lower frequency work is essentially the remainder of the workload. One improvement that most probably will be made in the future is to solve for altitude in the fast

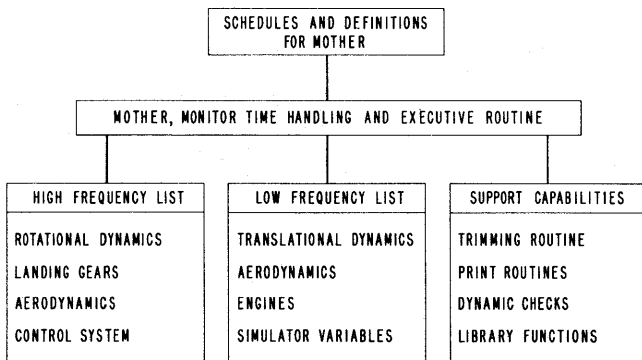


Figure 8—Organization of the digital simulation program

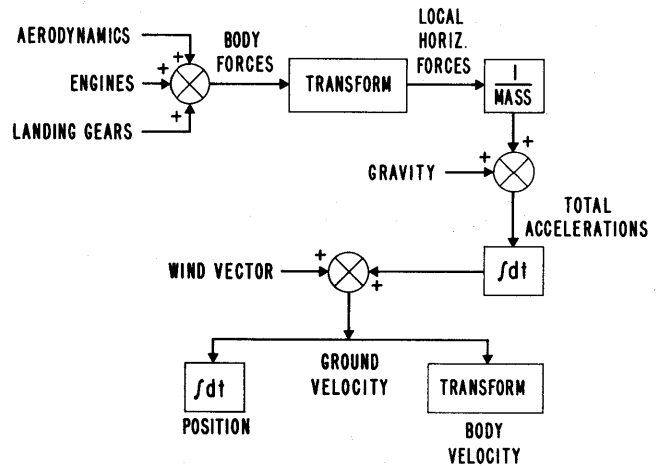


Figure 9—Translational dynamics block diagram

loop for improvement in the simulated landing gear response.

The digital simulation depends upon several sub-routines some of which do not run in real time. Data printout and aircraft trim calculations cannot run in real time. However, the three dynamic check routines which provide modal response checks must run in true-time. These routines are simply called by pressing a computer console push button. Other support software such as wind turbulence models, random number generators, and arbitrary functions of one, two, or three variables require a Fortran call in either the high or low frequency simulation loops.

Equations of motion

The equations of motion chosen for this simulation are a six-degree-of-freedom rigid body set. The set assumes a flat non-rotating earth in which the linear accelerations are integrated in a local horizontal Euler axis system and the angular accelerations are integrated in the vehicle's conventional body axis. This set of equations is sufficient for landing studies in which the range traversed is only four or five miles or less and the maximum velocities are very low, 60 to 150 knots.

The translational equations are interesting in that all forces, not including gravity, are summed in the body axis frame then transformed to the local horizontal axis where gravity is easily added in before integrating to obtain ground velocities. Wind velocities may be easily inserted in this axis system in the north, east, and down directions. No resolution of winds or gravity from Euler axis to body axis is necessary in this formulation. This is illustrated in Figure 9. One advantage of

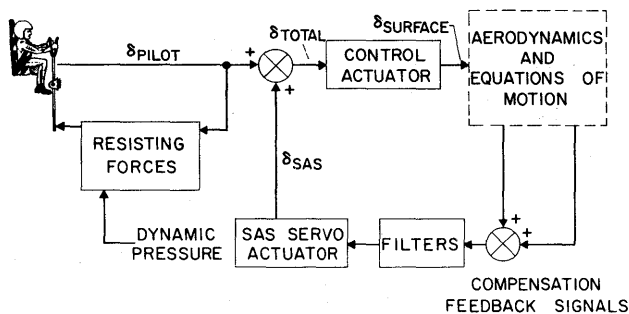


Figure 10—Typical control system and stability augmentation system

integrating the forces in the inertial frame is that the $\omega \times \bar{x}$ terms present in body axis acceleration equations are omitted along with the corresponding higher frequency content in the angular velocity terms. In digital simulations this is to be desired as reduction of frequency content usually improves solution accuracy.

Aerodynamics

The aerodynamic equation set formulates all the effects of velocity, control deflections, etc. and produces three forces and three moments. The stability derivatives were formulated from wind tunnel data taken with respect to the stability axis; from which a rotation through the angle of attack, α , about the y axis produces derivatives in the body axis frame. The equations and data are representative of many simulations with just a few interesting exceptions. The effects due to angle of attack of the horizontal tail takes into account the downwash of air flow over the wings onto the tail and the variable time delay involved between the distance from wing to tail as a function of speed. The most interesting facet of the aerodynamics simulation is the separation of the effects of right and left ailerons, due to simulated engine failures which stop the air blowing on an aileron with the resultant loss of aerodynamic control force.

Two types of random disturbances were included in the simulation. The first type was a "wing drop" in which a roll acceleration was inserted for a specific period of time to produce a resultant roll angle. The second was a wind gust model which produced noise in the three rotational and three translational velocities using a Dryden model.

Control systems and stability augmentation systems

Figure 10 is a block diagram of a longitudinal control system. This figure shows "force feel" modification

under computer control as a function of dynamic pressure and control column movement (as shown in the figure) or by surface deflection. It also illustrates how the control actuator, in the forward control path, and the SAS, in the feedback control loop, serve to affect the amount of control surface deflection obtained by column movements. In each control mode, lateral, longitudinal, and directional, there is a similar control system to provide correct pilot work loads.

In the actual aircraft the longitudinal control system is a purely mechanical system while both the lateral and directional systems have hydraulic power assist actuators. Provisions are made in the simulation to fail the hydraulic systems with resulting losses in aileron, spoiler and rudder effectiveness. In this vehicle stability augmentation is necessary in lateral and directional modes only. The lateral SAS system uses sideslip angle, roll rate and yaw rate feedbacks for roll stabilization while the directional SAS uses roll rate and sideslip rate feedbacks for stabilization in yaw. Wherever possible the linear filters are mechanized as simple difference equations using state space transform methods.²

Engines

The engine simulation is primarily a thrust computation in which the port and starboard engine thrusts are calculated separately from their individual throttle and diverter control levers. The engine thrusts contribute to body axis forces and moments for the aerodynamic computations. The jet engines produce hot thrust for propulsion and cold thrust for the flaps and ailerons.

The engine diagram, Figure 11, shows the basic ideas of the thrust simulation. The operation of this circuit

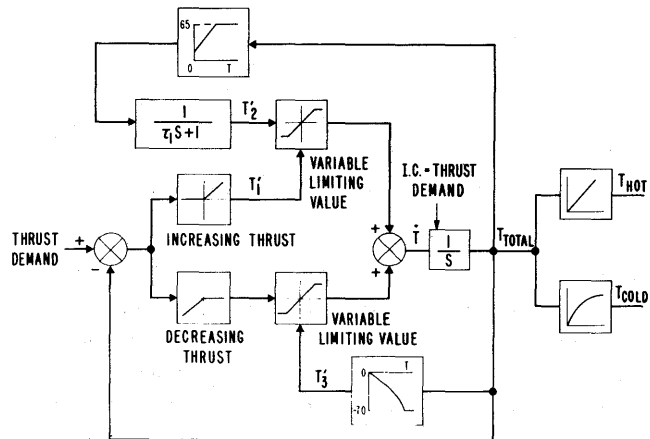


Figure 11—Fan jet engine thrust block diagram

is based on non-linear rate limiting of a first order system. When a demand for increased thrust is made the thrust rate, T , is either T'_1 or T'_2 . Initially T'_1 is greater than T'_2 , so no limiting occurs, and the limited thrust, $P_L < 65$. Thrust, T , is a positive exponential function of time but as thrust builds up and P_L becomes limited, the thrust will become linear with time until $T'_1 \leq T'_2$. At this time the limiter acts to make $T = T'_1$ providing a first order exponential tail-off of the process. The rate limiting used when thrust is decreasing is much simpler; the maximum thrust rate bares simply a square relationship to thrust. This scheme provides high thrust rates at high thrust levels and quite low thrust rates at low thrust levels.

The thrust diverter dynamics are modelled with rate limiting and hysteresis between the control and diverter angle. For determination of pilot handling qualities, system failures were implemented providing thrust loss in either of the two engines and a "hard over" diverter lock to a specific angle.

Landing gear

The landing gear model is composed of equations for tire friction forces and oleo reaction forces and the proper resolution of the forces into body axis forces and moments. The friction forces are resolved into two components, one in line with the tire and the other perpendicular to it. Coefficients of friction for both components are functions of gear velocities. The equations on gear compression and compression rate are rigorous but assume no tire deflection, so all reaction forces are due to the oleo. In the case of the C8-A three individual gears were simulated with the forces and torques on each summed to provide the total landing gear force and moment components.

Subroutine ICTRIM

ICTRIM is a subroutine which performs two separate functions. "IC" refers to the calculation of Initial Conditions (ICs) for the velocity terms in the local horizontal or Euler Frame based on inputs of airspeed, V_a , sideslip angle, β , and angle of attack, α . "TRIM" refers to the capability of this subroutine to trim the aircraft longitudinally.

In the operation of simulations at Ames it has been found that while use of the Euler frame to integrate accelerations improved calculation accuracies it did pose operational problems. In the use of the simulations it was apparent that research people using the program were more used to thinking in terms of total airspeed, for instance, than its components in body axes much

less those in Euler axes. Consequently a simple routine was written to accept V_a , α , β to calculate initial conditions for north, U_E , east, V_E , and down, W_E velocities. Later it was modified to allow the flight path angle, γ , to be input along with α to calculate the initial condition of pitch angle, θ .

Starting the aircraft simulation run with a trimmed aircraft avoids requiring the pilot to waste time trimming the aircraft before starting his task. For a large class of simulation problems the trimming need only consist of nulling pitch acceleration and the aircraft longitudinal and vertical accelerations. When aileron, rudder and sideslip angles and the roll and yaw rotational velocities are zero, the remaining three accelerations are zero.

The trimming algorithm is an iterative scheme. In the routine's most conventional form elevator control, δe , is used to null pitch acceleration, \dot{q} ; thrust, T , is used to null longitudinal acceleration, A_x ; and angle of attack, α , is used to null vertical acceleration, A_z . As an example of the iterative process the current value of \dot{q} is used to modify the current value of δe according to the equation $\delta e_{i+1} = \delta e_i + k\dot{q}_i$ where k is an appropriate gain predetermined from the aerodynamics of the aircraft. After modifying δe the routine commands a cycle of calculation through all the aircraft equations and data using the new value of δe to obtain a new value of \dot{q} . Of course α and thrust are being changed concurrently in the same manner. After sufficient iterations and for reasonable initial condition the routine determines the control inputs for which the accelerations are sufficiently close to zero for the airplane to be considered trimmed.

This basic scheme has proved to be sufficient for transport aircraft. However the C8-A has the capability of directing its thrust from 18° to 116° from the aft horizontal and when using this thrust vectoring to obtain trim the throttle may be held constant so that the diverter angle then becomes the trim parameter affecting both A_x and A_z strongly. A more general scheme was devised for the iteration algorithm since thrust diverter angle influenced both vertical and longitudinal accelerations. Basically the chain rule of differential calculus was employed to introduce the effects of α and thrust diverter angle, ν , on A_x and A_z . We say that

$$\Delta A_x = A_x - A_x^{\text{desired}}$$

but since the desired A_x is zero then

$$\Delta A_x = A_x$$

By the chain rule

$$\Delta A_x = (\partial A_x / \partial \alpha) \Delta \alpha + (\partial A_x / \partial \nu) \Delta \nu$$

In like manner

$$\Delta A_z = (\partial A_z / \partial \alpha) \Delta \alpha + (\partial A_z / \partial \nu) \Delta \nu$$

This is a set of two equations in two unknowns if we assume the partial derivatives can be found from the aero and engine data. Now we say that

$$\alpha_{i+1} = \alpha_i + \Delta \alpha \quad \text{and} \quad \nu_{i+1} = \nu_i + \Delta \nu.$$

where $\Delta \alpha$ and $\Delta \nu$ are the solution of the two chain equations. The partials of A_x and A_z with respect to ν are merely thrust modified by the sines and cosines of the diverter angle, but the derivatives with respect to α are unknown analytically and difficult to determine exactly.

The approximation $\partial A_z / \partial \alpha = \Delta A_z / \Delta \alpha$ was implemented by having the trim routine determine A_x and A_z for some α_i and again for α_i plus one degree then using those results to calculate the approximations. Since the approximation to the partial derivative was made rather arbitrarily the calculated $\Delta \alpha$ and $\Delta \nu$ were multiplied by a constant less than 1 to assure convergence. Most generally .8 was used, for instance, $\alpha_{i+1} = \alpha_i + .8 \Delta \alpha$.

This scheme in all other respects acts as the original more crude iterative scheme for conventional aircraft trim, iterating until the three accelerations are nulled.

Dynamic check routines

The subroutine DYNCHK is used to perform dynamic checks of the aircraft. The routine provides doublets and pulses in roll control and rudder control. In elevator control the optional disturbances are steps and pulses. The disturbances are input as pilot control variations providing a check on the control system as well as the aerodynamics. By recording the response data on eight channel strip chart recorders the user may view the results and determine the parameters in which he is interested.

Print routines

Two separate print routines are included in the simulation. One is a general purpose print routine for determining the status of a list of variables and is useful for printing initial conditions and trim conditions for documentation as well as a trouble shooting aid. The second routine is an attempt to determine the pilot's and aircraft's performance. It collects data in two ways as functions of altitude. Variables such as airspeed and

climb rate are saved at predetermined altitudes while the maximums of such variables as pilot control forces and guidance errors are determined within certain altitude ranges. If this routine is desired it automatically prints after completion of the run. This type of data is useful to correlate with the pilot's subjective ratings of the vehicle's handling qualities.

Support subroutines

Three subroutines from the computer library should be mentioned in the context of supporting simulations. They are WIND, MLTPLX and INSCAL. The latter two were described as hardware support routines. Probably the most important is WIND since it provides the atmospheric turbulence needed in simulations for aircraft handling qualities work. Turbulence is used primarily to assess the effects that real-life turbulence has on controllability, flying qualities, and ride qualities of an aircraft. The disturbance effects on the design of controls and stability augmentation systems are very important to insure that the airplane has sufficient control effectiveness to be manageable during flight in turbulence.

The turbulence model is the Dryden model.^{3,4} In essence white noise is passed through filters to provide noise in the three translational and three rotational velocities which have good representations of the power spectral densities present in actual air turbulence.

EPILOGUE

The simulation program and simulator hardware provided test pilots a realistic representation of the modified C8-A Buffalo with the result that various control and SAS representations were evaluated using pilot handling qualities ratings.

Final design parameters were found for the aircraft which is scheduled for flight test in early 1972. The digital program has subsequently been used as a base for additional simulations of navigation and guidance of STOL craft in the air traffic control situation near airports and for studies of control and SAS in longitudinal motion for this class of airplane. Of current popular interest are the noise reductions made possible by the high angle landing approaches to the runway which in turn are made possible by the slow approach speeds. At any ground point the STOL aircraft will be at a higher altitude than conventional aircraft thereby reducing the noise.

Due to the diverse uses of the program it is not con-

sidered to be fixed as present effort is aimed at determining better simulation techniques and models and at making the system software execute faster and be more responsive to the needs of the user.

REFERENCES

1 E A JACOBY J S RABY D E ROBINSON
FAMILY I: Software for NASA-Ames simulation systems
AFIPS Conference Proceedings Vol 33 Part 1 1968

2 J V WAIT

State-space methods for designing digital simulations of continuous fixed linear systems
Transactions of IEEE/PGEC Vol EC-16 No 3 1967

3 F NEWMAN J D FOSTER

Investigation of a digital automatic aircraft landing system in turbulence
NASA TND-6066 1970

4 C R CHALK

Background information an user guide for MIL-F-8785B (ASG), "Military specification—flying qualities of piloted airplanes"
AFFDL-TR-69-72 1970

Software validation of the Titan IIIC digital flight control system utilizing a hybrid computer

by R. S. JACKSON and S. A. BRAVDICA

Martin Marietta Corporation
Denver, Colorado

INTRODUCTION

In April 1966 work was initiated by Martin Marietta Corporation (MMC), Denver Division, to extend the role of an airborne computer to include flight controls as well as guidance and navigation computations. This project is one of several significant improvements for the Titan IIIC space booster which was funded by the Space and Missile Systems Organization of the Air Force. The new **Digital Flight Control System (DFCS)** has been successfully tested in four (4) Titan IIIC missions.

The purpose of this paper is to describe how a large hybrid computer simulation was used as an aid to design and develop the DFCS and then used to validate the resulting DFCS airborne software.

The simulation was programmed in six-degrees-of-freedom and included an airborne Univac 1824M **Missile Guidance Computer (MGC)** in the closed loop. Additional computing equipment used in the simulation included three (3) EAI 8800 analog computers, an EAI 8400 digital computer and an SDS 930 digital computer. Flight control hardware components such as rate gyros, body mounted accelerometers, and hydraulic actuators were also used in the simulation.

DESCRIPTION OF THE VEHICLE AND A TYPICAL SIMULATED MISSION

The Titan IIIC is one of the Titan models being built to carry Air Force and Defense Department payloads. The Titan IIIC is a four-stage vehicle with solid-propellant five-segment rocket motors on each side of a liquid propellant core. Stage 0 is powered by dual solid-propellant engines; Stages I and II are powered by gimbaled liquid propellant engines; and Stage III (transtage) is powered by a pair of restartable gimbaled liquid propellant engines. The transtage also has 12

mono-propellant attitude control engines, which are used for control purposes during the non-powered flight (coast) portions of a mission.

The Titan IIIC is required to handle single and multiple payloads that range from 1800 pounds to 30,000 pounds. There is also a variety of missions, one of which is described below, to match the spectrum of payloads. A wide range of payloads, in turn, significantly impacts the upper stage total vehicle mass, inertia, and dynamic characteristics which results in a requirement for flexibility in control system compensation. While in coast, the **Digital Attitude Control System (DACs)** must provide several levels of pointing accuracy as well as the capability to optimize time response and propellant utilization in a variety of inertia conditions. Furthermore, the powered flight Titan IIIC booster characteristics present to the control system designer a multi-plant problem with significant structural bending and propellant slosh dynamics. One of the DFCS design goals was to provide the capability to fly all of the missions and payloads with one basic airborne software package. Therefore, when the mission or payload change only modifications to the program parameters will be required.

A diagram of the particular mission that was simulated is shown in Figure 1. A 92-by-109 mile parking orbit is achieved by the boost portion of the flight which includes a 17 second 1st burn of transtage. After a one hour coast period, transtage will then fire for the 2nd burn, lasting 298 seconds, to obtain an elliptical transfer orbit of 107-by-22,300 miles. After about five hours in this path, the transtage will perform the 3rd burn to inject the satellite into a near-circular synchronous orbit measuring 22,221 miles at perigee and 22,318 miles at apogee. This typical synch-eq mission requires about 6½ hours from liftoff to satellite eject. (One of the software validation runs is a 6½ hour continuous run which duplicates this mission.)

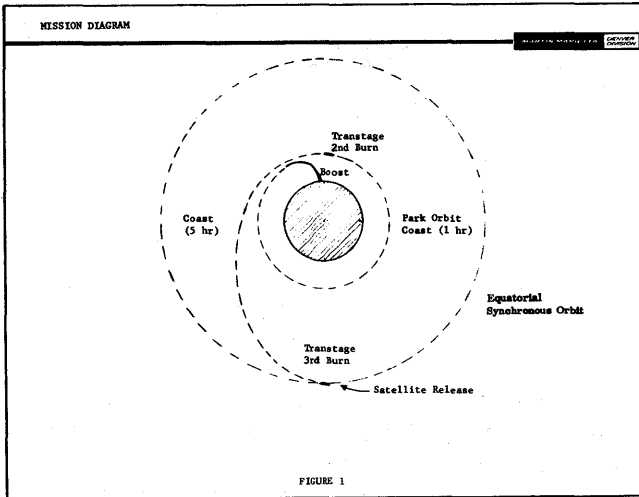


Figure 1—Mission diagram

During the coast portions of transtage flight, the simulation of the digital attitude control system (DACS) is required to perform a series of maneuvers to maintain proper thermal balance of the payload and to position the transtage for transmission of telemetry signals. In addition, the DACS is required to operate for several seconds in one or more velocity vernier modes wherein the eight aft-pointing DACS engines are turned on to make fine adjustments in the trajectory and to bottom the propellants in preparation for a main engine burn. This mode requires that the attitude control logic on these eight DACS engines be reversed because attitude control is maintained by turning a jet off in this mode rather than on, as in normal operation.

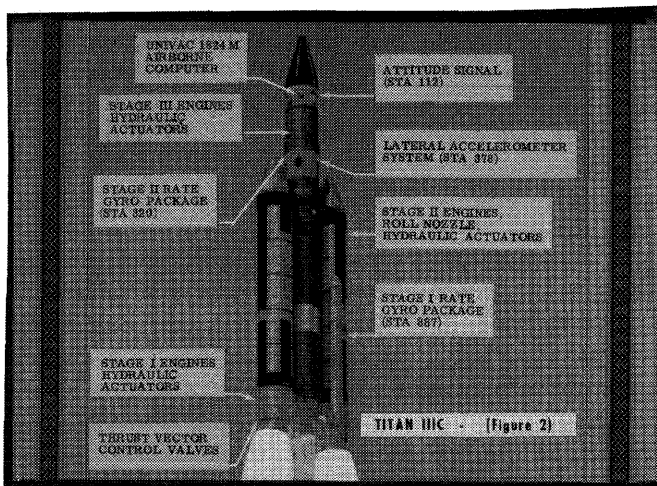


Figure 2—Titan III C configuration

A diagram of the Titan III C is shown in Figure 2, and the pertinent flight control hardware that was used in the closed loop simulation is labeled.

DESCRIPTION OF THE HYBRID SIMULATION

Since flight hardware was included as part of the simulation effort, the simulated airframe had to be computed in real time. The real time computational requirement and the size of the simulated airframe dictated the need for a hybrid simulation. The simulation hardware as shown in Figure 3 is located in two separate facilities; the hybrid computation facility and the controls mockup facility. The distance between the two facilities is approximately 300 ft.

Facilities description

The hybrid facility contains 3—EAI 8800 analog computers, 1—EAI 8400 digital computer with a 32 bit per word 32K memory, and a linkage system containing 32 analog to digital converters and 32 digital to analog converters.

The Controls Mockup Facility (CMU) contains the Univac 1824M missile guidance computer, an SDS 930 digital computer, flight actuation devices for all stages, flight hardware sensor devices, and interface equipment for buffering the signals received from the hybrid facility. Figure 4 is a photograph of the inverted engine bells which are driven by hydraulic actuators with Stage II in the foreground, Stage I in the middle and transtage in the background. Engine commands gener-

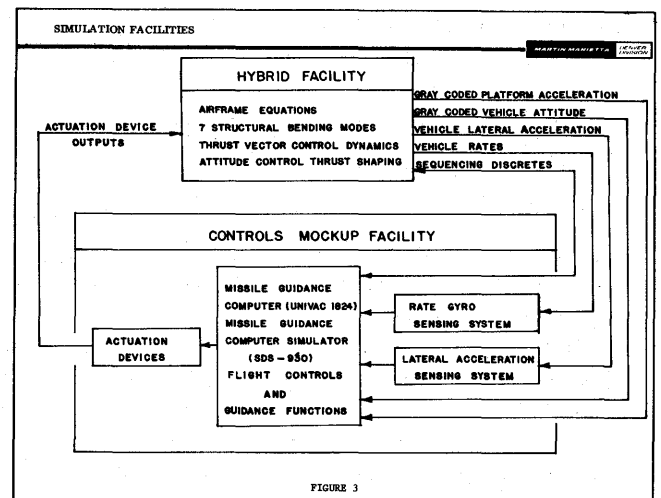


FIGURE 3

Figure 3—Simulation facilities

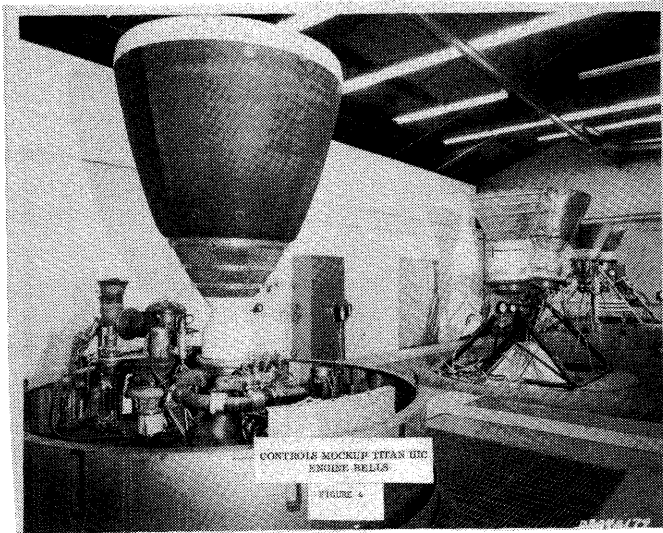


Figure 4—Controls mockup Titan IIIC engine bells

ated in the MGC were output to the actuation devices. Engine displacements measured by telemetry potentiometers on the actuation devices were used to provide feedback into the airframe simulation.

The Univac 1824M missile guidance computer is a binary machine employing fixed point, two's complement arithmetic with single address capability. The MGC employs a non-destructive readout thin-film memory which can store 12,096 16-bit words.

The SDS 930 digital computer contains a 16K memory with 24-bits per word. Prior to using the MGC in the closed loop simulation, the digital flight control system, while still in the development stage, was programmed on the SDS 930 computer. The SDS 930 computer was then utilized to aid in checkout of the airframe simulation and also to provide a means of easy access for investigating design considerations for the DFCS. The final MGC software was translated for use in the SDS 930 computer, therefore allowing the SDS 930 to be used as a missile guidance computer simulator (MGCS). The primary purpose of the MGCS was to stand backup for the MGC during the critical DFCS software validation period.

The hybrid simulation

The airframe simulation required the use of 3—EAI 8800 analog computers utilized to 95 percent of their operational amplifier capability and a digital program requiring 16K of core in the EAI 8400 digital computer. The assignment of computational tasks to the analog and digital computers was handled in such a manner as to make use of the best computational aspects of

both computers. The parts of the simulated airframe mechanized on the analog computer were as follows: The body acceleration and body rate equations; seven system modes representing structural bending and fuel slosh for all four stages; vehicle sensor station equations; a Thrust Vector Control system (TVC) for the solid rocket motors of Stage 0; attitude control system thrust shaping, and an analog autopilot to aid in the checkout of the simulated airframe. Two-bit Gray coders were implemented on the logic panels for the Gray coding of the simulated inertial platform accelerations and vehicle attitudes. Functions used in the thrust vector control system were generated on card programmed diode function generators. All other functions required in the simulation were generated in the 8400 digital computer.

Resolution on the body rate and body acceleration equations was maintained by releveing the input functions originating in the digital computer at vehicle staging events. The simulation of the seven system modes required the generation of 6 functions per mode per stage; therefore, a total of 168 separate functions were needed in the simulation of the seven system modes for a complete mission. Again the digital computer provided a means of generating the required functions and also supplied a method of rescaling at staging times, which made it possible to conserve on the amount of analog equipment required to simulate the seven system modes.

On the actual vehicle, platform acceleration and vehicle attitude are generated by means of optisyns which produce a two-bit Gray code as output to the missile guidance computer. Both platform acceleration and vehicle attitude in the simulated airframe were computed in the EAI 8400 digital computer. Vehicle attitude was updated every 10 milliseconds. The attitude was then quantized and compared with the previous pass. If a change of one quanta in attitude was observed, the information was sent to the analog patch panel for Gray coding and then output to the missile guidance computer. Gray coding of platform accelerations was handled in the same manner as vehicle attitude except that they were updated and output every 20 milliseconds.

The digital portion of the simulated airframe consisted of two routines. Routine 1 was updated every 10 milliseconds in Stages 0, I, and II and every 5 milliseconds during transtage flight; routine 2 was updated every 20 milliseconds throughout the mission. Using the external interrupts of the EAI 8400 digital computer, routine 1 was given higher priority than routine 2. The timing for the interrupts was generated on the analog logic panel via a binary coded decimal down counter. Routine 1 sampled the body rates generated

on the analog computer and utilized the body rates in the generation of the direction cosine matrix. Vehicle attitude was generated as a function of direction cosine terms, and then quantized and output to the analog logic panel for Gray coding.

Routine 2 sampled the body accelerations and, using the direction cosine matrix generated in routine 1, transformed the body acceleration into inertial acceleration. Subtracting the gravity component from the inertial acceleration and integrating produced inertial velocity. Another integration produced inertial position. A transformation on the inertial velocity using the inverse direction cosine matrix produced the body velocities from which the aerodynamic terms were formed. The aerodynamic terms were then output to the analog computers for input to the body acceleration and body rate equations. Platform acceleration was computed, quantized and output to the analog logic panel for Gray coding. All functions generated in the digital computer were computed and output in routine 2. An Automatic Data Channel Processor (ADCP) was used to store digital variables on magnetic tape at 1-second intervals. The tape was then post-processed to retrieve the real time information.

The flight computer generates timing and sequencing discrettes which control staging, engine start and engine shut-down commands. These discrettes were sensed in routine 2 and the necessary action was initiated on the simulated airframe.

Simulation checkout

The general purpose hybrid facility was scheduled in 6-hour shifts; therefore, setup of the simulation airframe was a daily occurrence. Potentiometer settings and static tests were performed with each setup of the simulation, using the Hytran Operations Interpreter which is an EAI processor designed to perform checkout functions on the analog equipment. To further insure that the airframe simulation was operating properly, a trajectory was flown using an analog autopilot. This trajectory run was made daily before closing the loop around the missile guidance computer and all the associated flight hardware.

In addition to the static checkout method, a thorough dynamic checkout procedure was developed to ensure that the airframe simulation represented the physical plant. Part of this procedure required that the entire closed loop simulation be in a perturbate mode. This is a mode wherein the trajectory is fixed in time; thus all vehicle parameters are held constant.

The missile guidance computer has a perturbate mode built into the software. This mode allows the

autopilot gains and digital filter parameters to be held constant. Also the guidance commands into the autopilot are held constant and all guidance computations are bypassed. When the missile guidance computer is put into the perturbate mode, a discrete is issued to the hybrid simulation. The perturbate discrete actuates logic incorporated in the EAI 8400 digital program which holds all parameters that are functions of time and also holds vehicle altitude. The discrete also sets the longitudinal body acceleration in the analog computer to zero.

The dynamic checkout procedure consisted of two main parts. The first part included frequency response testing with the simulation in the perturbate mode. Frequency responses of the closed loop simulation were generated by forcing engine deflections with a sinusoidal signal. In this way, stability margins at several time points in each stage were verified against previously generated stability analysis results. The second part of the dynamic check involved comparing the hybrid simulation trajectory results with the results of an all-digital trajectory program.

DESIGN AND DEVELOPMENT OF THE DFCS

A new set of analytic, software, and simulation problems were generated by incorporation of a DFCS into the Titan IIIC. This section of the paper describes how some of these problems were resolved with the aid of the hybrid simulation.

Digital filter accuracy

Figure 5 is a block diagram of the Stage 0 pitch plane autopilot, showing the four feedback loops used. The attitude and two angular rate loops are used only for stabilization purposes, while the lateral acceleration feedback loop is employed as a means of achieving active load relief, and is operative only during the max

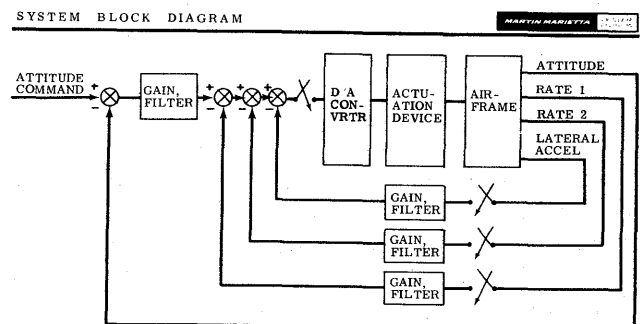


Figure 5—System block diagram

buffet-max dynamic pressure portion of Stage 0 flight. Each of the feedback loops is compensated with a gain and a digital filter.

The digital filters are susceptible to accuracy problems¹ when mechanized on a fixed point digital computer such as the Univac 1824M. Two main sources of the accuracy problem are (a) difference equation coefficient accuracy, and (b) fixed point digital computer truncation accuracy. Furthermore, the requirement that the software design will be able to fly the wide range of payloads and missions without the reprogramming of equations makes the accuracy problem all the more acute. For example, filter computations must be scaled in such a way as to insure no overflow for large inputs for all possible combinations of mission/payload. Then for low level inputs (such as those expected for nominal operation) computations are scaled in a very sub-optimal manner, thereby exaggerating even further a computational truncation inaccuracy problem.

To determine the effect on system operation in regard to digital filter computational accuracy, a proposed DFCS was programmed on the SDS 930 digital computer. System operation in this case means the effect of interaction between filter accuracy and other vehicle characteristics such as quantization of MGC input and output variables, actuator nonlinearities, sensor nonlinearities, and propellant slosh. This possible interaction could cause excessive limit cycle amplitudes and even instability. Therefore, to verify the DFCS design, the DFCS was programmed on the MGCS and the operational word length was varied in the digital filters for each stage. This was done by "masking" bits in the word that are furthestmost to the right, thereby effectively varying the scaling (binary point location).

Noise susceptibility

Digital autopilot noise susceptibility is caused by the sampled data folding phenomenon. Relatively high frequency environmental noise brought into the system as corruption on the sensors will, upon being sampled by the DFCS, be "folded" to a lower frequency (i.e., any signal whose frequency, w , is greater than half the sampling frequency, $w_s/2$, will, on being sampled, be "folded" about $nw_s/2$, $n=1, 2, 3, \dots$). Hence, energy that was filtered out by an analog autopilot now appears in the form of low frequency signal content in the control system. This low frequency energy can cause excitation of the structural bending modes of the vehicle, thereby inducing significant structural loads. The severity of the loads problem is a function of the

amplitude and frequency content of the noise coming in on the sensors and of the sampling frequency of the DFCS.

To determine the severity of this problem on the Titan IIIC DFCS (sampling frequency of 25 samples per second), the mean and 3σ environmental noise spectrum existing during flight was measured. This was done by statistically reducing telemetry data on sensor outputs from several past Titan IIIC flights which used analog autopilots. Next, a noise generator on an EAI 8800 console plus shaping networks were set up to duplicate the in-flight environmental noise. When the 3σ noise was superimposed on sensor outputs, and therefore sent directly into the autopilot, excessive excitation of the first structural bending mode was seen and intolerable vehicle loads were generated. This problem was solved by using analog prefilters (by filtering the sensor input to the MGC) with a break frequency of 10 cps, .5 damped on the rate channels and a 5 cps break frequency, .5 damped on the lateral accelerometer channels. These prefilters were specified to be built into the MGC and are common to all flights.

Recursive and non-recursive digital prefilters were derived and tested by using the MGCS. However, significant noise energy proved to be present at frequencies above $w_s/2$ which, due to the folding phenomenon, rendered the digital prefilters much less effective than analog prefilters.

An extremely useful application of the EAI 8400 digital computer was made in conjunction with the noise investigation, wherein a program was written which would sample the noise-shaping network outputs and calculate the power spectral density and rms level of the noise. Thus the validity of the noise simulation was easily and rapidly checked against desired results. This one application saved many hours that would have otherwise been spent waiting for the same results to be calculated on an "off-site" digital computer.

Digital autopilot malfunction detection logic

Experience with airborne computers indicates that the prevalent failure mode is a transient one. In this failure mode an electrical transient can cause either read-write memory or one of the central processor registers to pick up or drop one or more bits of information. The result is that the program will transfer incorrectly, perform an incorrect calculation, or store bad data.

As part of the DFCS development, MMC designed malfunction logic that will detect these types of errors and reset the MGC to a safe-point condition. The

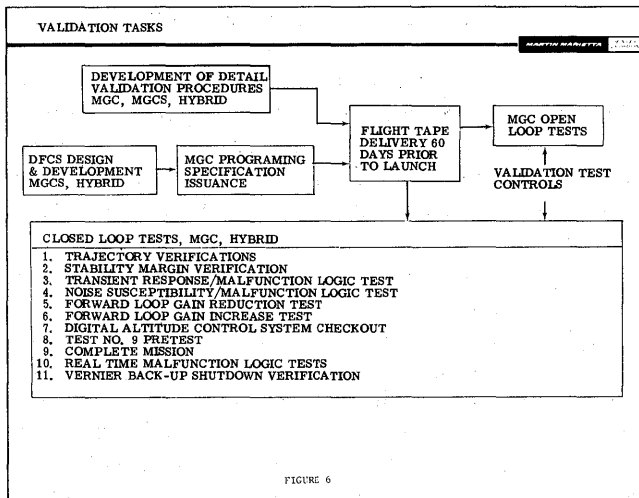


Figure 6—Validation tasks

safe-point condition means that all possible flight control parameters are initialized from nondestructive readout memory. As implemented, 98 percent of the DFCS parameters are initialized. The key variables (stage indicators, time, and certain other key flags) which cannot be initialized are specially protected, decreasing the chance of their being altered. The overall malfunction logic requires approximately 8 percent of the total DFCS memory requirements.

The MGCS and the hybrid simulation were used to verify the operation of this system. Possible types of malfunction were simulated in the MGCS and the response of the flight was monitored in the hybrid lab. Only 40 milliseconds are required to complete the initialization process. The resultant transient to the vehicle, even in the maximum dynamic pressure region of flight with worst case winds, is almost undetectable.

SOFTWARE VALIDATION

The software is in the form of a flight tape—a punched paper tape that was coded to MMC specification by Univac and delivered to MMC and other agencies for validation. Program validation has two main objectives. The first is to insure the correctness of equations, logic, timing, memory utilization, module interaction, etc., for the flight tape programming when compared to the software specification. The second objective is to insure that the program on the punched tape satisfies all of the mission and vehicle performance requirements. The validation itself consisted of a series of open and closed loop tests executed per a detail

procedure document. The open loop tests were an important part of the tape validation and consisted of phasing tests, open loop digital filter response, etc. However, these tests did not include the hybrid simulation and they will not be discussed further. The closed loop testing basically consisted of determining if the flight tape could “fly” the entire mission and meet all requirements.

Figure 6 illustrates the tasks that were necessary for software validation. One of these tasks required that a validation procedures document be written. This document was developed into a 537 page volume that detailed the test set up, the test procedure, and the expected results for each validation test. The expected results are called success criteria. Tolerances were placed on the success criteria in such a way that if validation test results should exceed these tolerances then the test is considered to be invalid. The reason for the test being invalid must then be traced and explained. The MGCS, MGC, and the hybrid simulation were used to develop and check out the tests, and in some cases, to generate success criteria. Development of the validation procedures document required approximately five man-months of effort, but this document proved to be invaluable during the flight tape validation period.

The closed loop tests that were performed are listed in Figure 6. To illustrate how these tests were run, test 1 is discussed briefly. The following wording is taken from the validation procedures document.

Validation Test 1, Trajectory Verifications:

1. Test Objective—This will be a baseline trajectory run through Stage III first burn and approximately 5 minutes into coast flight. The overall objective of this test is to uncover quickly any major trouble areas that may exist. The specific purposes of this test are to verify proper preaiming filter initialization and e.g., tracking, and to verify dynamic stability during staging. (The preaiming filter is a forward loop filter in the control system that tracks the vehicle center of gravity. Also this filter is “initialized” just before a main engine is fired which causes the thrust vector to be pointed through the center of gravity.)
2. Configuration—The test configuration is the Hybrid Computation Lab (HCL)/CMU Closed Loop 6-Degree of Freedom Trajectory Simulation Configuration as illustrated in Figure 3. For this test, the Stage III thrust differential is removed for Stage III start, but is included for Stage III shutdown.

TABLE I—Success Criteria and Test Results for Preaiming Filter Operation

Flight Condition	Success Criteria	Test Results	Test Results
	Engine Deflection	(Test 1)	(Test 9)
	Pitch (deg)	Pitch (deg)	Pitch (deg)
Stage II Start Initialization Value	$-.027 \pm .019$	-.0336	-.019
Stage II Burnout	$-.018 \pm .095$	-.0573	-.055
Stage III Start 1st Burn Initialization Value	$-.24 \pm .05$	-.254	-.23
Stage III End 3rd Burn	$-.162 \pm .28$	N.A.	-.419

3. Test Sequence—

- a. Set up HCL and CMU to use trajectory run procedure with the telemetry (TM) ground station.
- b. Label HCL strip charts and the $x-y$ plotters and enable Automatic Data Channel Processor. Plot the aerodynamic variables α and q_{AT} on $x-y$ plotters.
- c. Null to zero any biases on the TVC valves, TM monitor pots, and upper stage actuator TM pots. This is done in Stage 0 with the hybrid simulation in the I.C. mode. Stage I actuators are biased to zero with the simulation in "perturbate" at approximately $T=110$ sec. Stage II and Stage III actuators are biased to zero with the simulation in "perturbate" in Stage I.
- d. Success Criteria.

Success criteria for this test is given in Table I. Table I is taken from the validation document for this test to illustrate how the success criteria is typically used. The test results shown in Table I were obtained from telemetry data output from the MGC. (This is the only way that real time data could be continuously extracted from the MGC during validation tests.)

The most significant table of success criteria for this test is not included in this paper because of space limitations; however, it will be discussed. This table was designed to measure trajectory variables against success criteria that were generated by an all-digital trajectory program. In this table, the following hybrid computer trajectory variables for ten flight times are measured against the success criteria: total velocity, velocity components, altitude, angle of attack, side-slip angle, aerodynamic pressure, and the product of the aerodynamic pressure and the total angle of attack. All of these variables must be within the success criteria tolerances for the results to be acceptable. Tolerances for this

test vary from less than 1 percent upward depending on the flight time and the particular variable that is being examined. One other table of success criteria (not shown) was generated for test 1, which consists of the maximum vehicle attitude rates that should never be exceeded during staging sequences.

Test 7, which is a complex checkout of the digital attitude control system, involved a special application of the EAI 8400 digital computer. The DACS has several different logic "stages" in each of the three autopilot channels. Consequently, to check every logic path in the DACS, 132 separate sub-tests were required.

As in some of the other tests, support of the TM ground station was necessary for test 7. This support involved recording DACS information on magnetic tape. Hence, it was important that test 7 be executed as rapidly and efficiently as possible. To achieve this goal, the 8400 digital computer was programmed to accept data for the 132 sub-tests from punched cards and execute the entire sequence of test 7. More specifically, the digital program simulated vehicle attitude motion which was then Gray coded on an 8800 analog computer. The resulting Gray code was sent to the MGC to exercise the DACS logic. The time needed to complete test 7 for one channel was one hour of continuous running. The other tests listed in Figure 6 were conducted in a manner similar to test 1; therefore, they will not be discussed.

During the validation period, configuration control was carefully maintained in both the CMU and in the HCL. For example, whenever reprogramming of the hybrid computer was required, or a pot setting changed, this information was entered into a configuration control log book along with the reason for the change. In addition, the analog patch boards and the EAI 8400 program were kept locked in a special cabinet when not in use. Responsibility for maintaining configuration control and security during validation tests was assigned to the lead engineers who were running the tests.

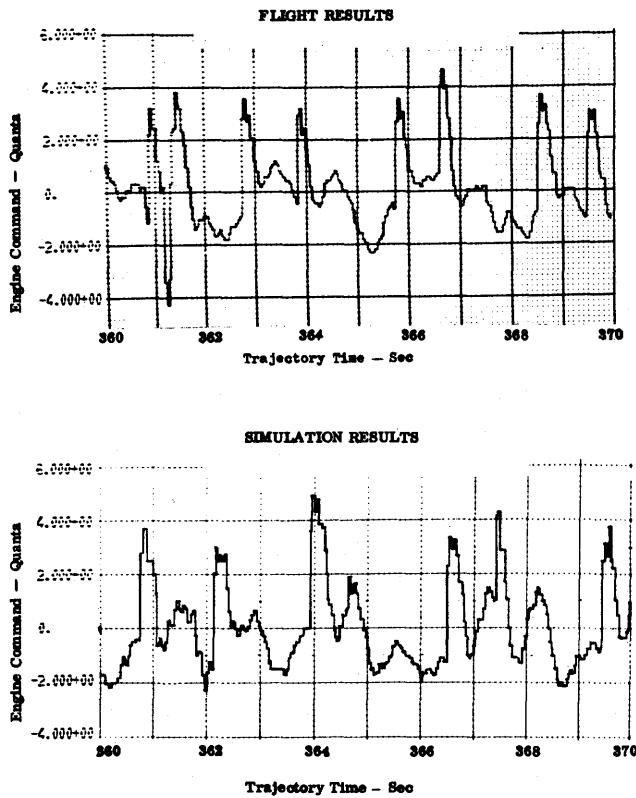


Figure 7—Simulation and flight result comparison

Figure 7 is a typical comparison of simulation results with results obtained from the flight. Pitch engine deflection commands are compared over a ten second time period during Stage II flight; trajectory time corresponds to 360 seconds at the beginning of the plots.

Each quanta of engine command is equivalent to .02 degrees of engine deflection. Figure 7 illustrates that the expected in-flight limit cycle results were substantiated quite well.

CONCLUSIONS

The hybrid simulation described in this paper proved to be a valuable aid in the development and design validation of a new digital flight control system for the Titan IIIC.

The simulation was instrumental in resolving problems associated with digital filter accuracy and digital flight control system noise susceptibility. Also malfunction detection logic for the airborne missile guidance computer software was developed. This developmental work contributed significantly to achieving an operational digital flight control system which is capable of flying the broad range of Titan IIIC missions without the necessity of reprogramming software.

The simulation was successfully used for validation of the final airborne software by executing a series of carefully planned validation tests. These tests were designed to verify performance of the entire digital flight control system.

REFERENCES

- 1 R S JACKSON
Minimization of computer word length and storage requirements in recursive digital filter design
IEEE Proceedings of the Fourteenth Symposium on Circuit Theory May 6-7 1971

Multivariable function generation for simulations

by S. P. CHEW, J. E. SANFORD and E. Z. ASMAN

Boeing Computer Services
Seattle, Washington

INTRODUCTION

The purpose of this paper is to describe the mechanization of a technique for generating continuous functions of up to six variables, given a discrete set of experimental data points. This technique is currently being applied in a real-time simulation of a high performance missile. The characteristics which make this method of function generation unique are the high speed with which many multivariable functions are produced and the large size of the data base from which the function values are computed.

The need for this capability emerged at a time when an existing hybrid simulation of the missile was to be improved and expanded to support missile flight tests. It was determined that in order to refine the simulation's predictive and post-flight analytic capability, a substantial improvement over the existing all analog method of generating the functions which define the aerodynamic model was required. With the all analog method, only a small portion of available data could be applied to model the missile aerodynamics. The objective was to develop a technique for direct utilization of wind tunnel data in order to provide a more accurate model.

The following discussion defines the problem and its solution constraints, considers alternative solutions and describes the mechanization of the selected approach. A summary of significant results achieved by the application of this technique concludes the discussion.

PROBLEM DEFINITION

The problem of generating aerodynamic functions in a real-time* simulation of a high performance missile is more difficult than in simulations of slower flight

* Real-time simulation is required in order to evaluate flight hardware in the loop and to collect sufficient data in a reasonable amount of time.

systems, primarily because of the higher system frequencies involved. The difficulties, however, become particularly acute when the number, shape and size of the missile control surfaces is highly restricted by carrier-missile interface design. The stringent performance requirements for maneuverability and the control surface design constraints, impose demanding requirements on the function generation task. Some important reasons for this are:

1. During certain maneuvers, a control surface (fin) may be flying in the shadow of the missile. The effectiveness of this control surface may be very non-linear with respect to the deflection angle.
2. The missile control system is only stable within a narrow region defined by the gain margin. Gain margin errors of a few db introduced by the function generation process may quickly invalidate simulation results.
3. The frequency response of the system is exceptionally fast. Any computational delay introduced has a pronounced effect on the gain margin.

In addition to all of these general constraints, the following specific simulation requirements were identified:

1. The number of functions to be generated in real-time, included:
 - 6 functions of 6 variables
 - 5 functions of 3 variables
 - 3 functions of 4 variables
2. A provision for a data base containing approximately 20 million discrete, wind-tunnel derived data points was required.
3. All function outputs were to be continuous to permit integration with the existing simulation

and to avoid degradation in gain margins. Unacceptable degradation was empirically determined to occur when function output delays exceeded 0.3 milliseconds.

4. The selected function generation technique was to provide the capability for rapidly altering functional data, as further wind-tunnel or actual flight test data became available.

A literature search of material related to multi-variable function generation and consultations with individuals and equipment manufacturers (See References 1, 2 and 3) yielded useful ideas which later influenced the design concept. Most notably, contacts with Mr. A. I. Rubin, author of Reference 3, provided the base from which the ultimate design evolved.

ALTERNATIVES

Several alternatives were evaluated for satisfying the above problem definition. Improvement to the existing all analog aerodynamic function generation system was rejected because lengthy and complex curve fitting techniques are required whenever changes to aerodynamic data points are made. Also, the system did not accurately represent the functions, due to inherent noise and inability to include sufficient data points. Evaluation of function generation using a general purpose digital computer revealed that unacceptable time delays would be created. It was determined that a Hybrid Function Generation System (HFGS) would best meet the stringent requirements for accuracy, flexibility of modifying function data points, and frequency response. Two mechanization alternatives, a multiplexed and a parallel, were evaluated. The multiplexed HFGS, which generated several functions with the same set of circuits, could greatly reduce the number of computing components. Although a reduction in components could theoretically be achieved, the component specifications would be extremely rigid and the resultant system would have limited general purpose applications. The parallel HFGS, which employed separate circuits for each function, could be designed with commercially available computing components and configured to meet the required multivariable function generation task as well as other simulation needs. Based on these evaluations, the parallel alternative was chosen for implementation.

The parallel HFGS (See Figure 1) was envisioned to consist of a large general purpose digital computer, interfaced with a special hybrid computing unit. The digital computer would be assigned the task of data

storage, management and input/output. The hybrid computing unit would perform the interpolation calculations in parallel, to produce continuous functions simultaneously.

HFGS DESIGN REQUIREMENTS

In order to meet the simulation requirements discussed earlier, certain internal design criteria had to be met:

1. The HFGS was required to produce updated function values in less than 5 msec. following a breakpoint* crossing of any one of the independent variables. Consequently, the internal update time was not to exceed 2.5 msec.
2. A correct set of 442 data points had to be retrieved from storage and made available for interpolation within the update time.
3. The data requirements had to be met within the storage constraints of the available digital computer.
4. The interpolated functions had to match within 1 percent the theoretical values obtained from linear interpolation of wind-tunnel data.

The stringent 5 msec. requirement was derived from considerations of missile frequency characteristics, minimum expected spacing between breakpoints and time delay constraints. A longer update time was expected to compromise the 1 percent accuracy specification considered necessary to provide realistic results.

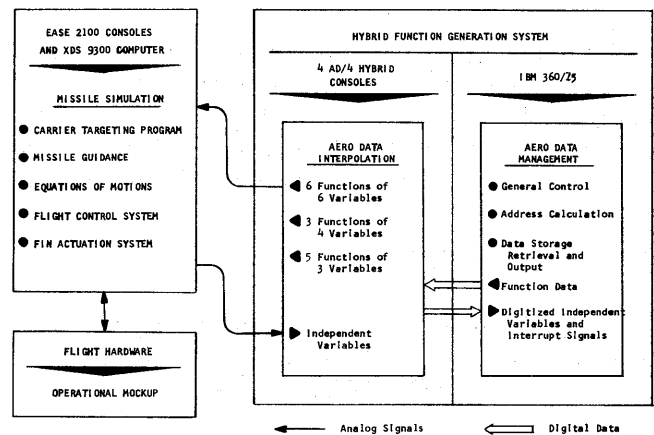


Figure 1—The Boeing hybrid function generation system

* Breakpoint is a predefined discrete value, X_i , of an Independent variable X for which a function data point $f(X_i)$ exists.

In order to provide for the occurrence of two breakpoint crossings in rapid succession, the internal update time had to be 2.5 msec. This would ensure that sequential processing of two closely spaced breakpoints would not exceed the allotted 5 msec. to accomplish a function update.

DATA REDUCTION TECHNIQUES

The HFGS was required to produce functions from data matrices of $17 \times 13 \times 13 \times 13 \times 13 \times 7$ (3,398,759 points), $17 \times 13 \times 13 \times 7$ (20,111 points) and $6 \times 3 \times 25$ (450 points) for functions of 6, 4 and 3 variables respectively. To represent the specified number of multivariable functions, the resulting data base would be 20.5 million points. Although required to accurately describe the highly non-linear aerodynamic functions, the data base was impractical to obtain from direct wind tunnel measurements or to fit into the random access memory of the available digital computer.

Fortunately, not all of the 20.5 million points were needed to reproduce an *EFFECTIVE* data base of the specified size.

The aerodynamic coefficients produced by the HFGS are functions of six independent variables: angle of attack, α , side slip angle, β , velocity, M , and the control surface deflection angles δ_1 , δ_2 and δ_3 . Although a six dimensional data matrix is needed to describe a function of six variables, in practice a minimum set of single fin effective data is obtained from wind tunnel measurements. The effect of each control surface is incrementally combined to produce an equivalent function of 6 variables. This superposition of the fin effects reduces a function of 6 variables to the sum of 3 and 4 variable functions. The superposition algorithm for this reduction is:

$$\begin{aligned} f(\alpha, \beta, M, \delta_1, \delta_2, \delta_3) = & f_0(\alpha, \beta, M) \\ & + f_1(\alpha, \beta, M, \delta_1, 0, 0) - f_0(\alpha, \beta, M) \\ & + f_2(\alpha, \beta, M, 0, \delta_2, 0) - f_0(\alpha, \beta, M) \\ & + f_3(\alpha, \beta, M, 0, 0, \delta_3) - f_0(\alpha, \beta, M) \end{aligned}$$

which simplifies to:

$$\begin{aligned} f(\alpha, \beta, M, \delta_1, \delta_2, \delta_3) = & f_1(\alpha, \beta, M, \delta_1) \\ & + f_2(\alpha, \beta, M, \delta_2) + f_3(\alpha, \beta, M, \delta_3) - 2f_0(\alpha, \beta, M) \end{aligned}$$

The $17 \times 13 \times 13 \times 13 \times 13 \times 7$ data matrix required for a function of 6 variables was reduced to one matrix of $(17 \times 13 \times 7)$, and 3 matrices of $(17 \times 13 \times 7 \times 13)$ for functions of 3 and 4 variables respectively. This reduces 3,398,759 data points to 61,880. Since redundant data at $\delta_1 = \delta_2 = \delta_3 = 0$ in these matrices can be eliminated, the

number of data points would be 57,239 for each of the 6 functions of 6 variables.

In addition to superposition, vehicle symmetry presented the opportunity to further reduce the data base. Function values corresponding to negative angles of an independent variable can be derived from data measured at positive angles. The HFGS is programmed to compute data points derived from symmetry during each update cycle.

The above techniques reduced the specified 20.5 m point data base to approximately 435,000 points.

The available digital computer was an IBM 360/75 equipped with a 750K byte core memory. The data base of 435K, 15-bit data points would require 870K bytes. Since the total data base could not reside in core, the total function data was segmented into several pages and stored on drum. Each page contained data corresponding to a breakpoint of the slowest changing variable. For linear interpolation data from two adjacent pages are required. Thus only four pages (two immediate plus one ahead and one behind) need to be in random access memory at any time. Since the paging variable is relatively slow, a new page can be transferred to core before the paging variable exceeds the boundaries described by the resident data.

The data base required to produce the functions could now reside in core and be manipulated by the software. The application of these data management techniques did not compromise original system specifications.

THE INTERPOLATION ALGORITHM

The linear interpolation algorithm was chosen to:

1. Provide direct correspondence between equation and computing elements for easy hardware implementation and fault isolation.
2. Allow systematic expansion from functions of one to n variables.
3. Eliminate operational restrictions (such as maximum slope and data spacing) because of hardware limitations.

The algorithm can be derived directly from the principle of superposition as follows:

Figure 2 shows a typical function of one variable. Figure 3 shows the region of interest. For linear interpolation, regardless of how the rest of the data points are distributed, the value $f(X)$ at any point $X_i \leq X \leq X_{i+1}$ is determined by $f(X_i)$ and $f(X_{i+1})$. The principle of superposition states that the value of $f(x)$ can be computed as the sum of the individual con-

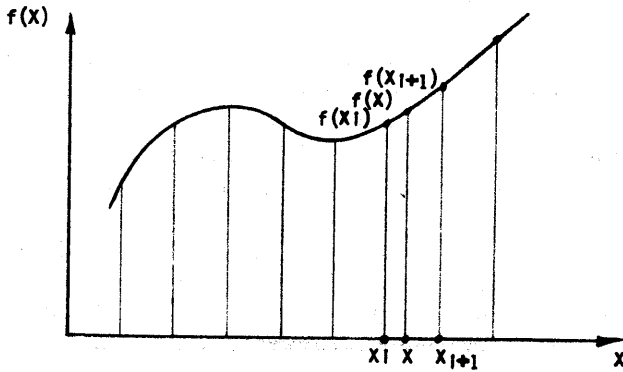


Figure 2—A function of one variable

tributions from $f(X_i)$ and $f(X_{i+1})$. The contribution of $f(X_i)$ is f_1 in Figure 2. From similar triangles:

$$f_1 = f(X_i) \left[\frac{X_{i+1} - X}{X_{i+1} - X_i} \right]$$

Similarly the contribution of $f(X_{i+1})$ is:

$$f_2 = f(X_{i+1}) \left[\frac{X - X_i}{X_{i+1} - X_i} \right]$$

And

$$f(X) = f_1 + f_2 = f(X_i) \left[\frac{X_{i+1} - X}{X_{i+1} - X_i} \right] + f(X_{i+1}) \left[\frac{X - X_i}{X_{i+1} - X_i} \right] \quad (2)$$

Although the spacing between data points is not equal,

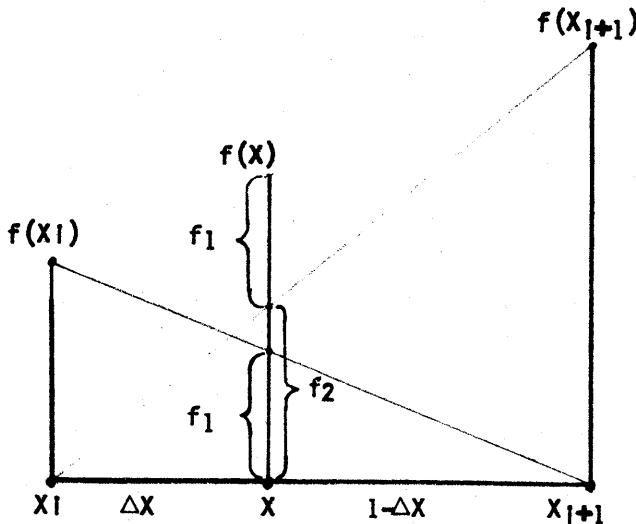


Figure 3—Generation of $f(x)$ within two known points by superposition

simplification of the equation is accomplished by normalizing each interval between X_i and X_{i+1} to unity. Thus, in Figure 2,

$$X_{i+1} - X_i = 1; \quad \frac{X - X_i}{X_{i+1} - X_i} = X - X_i = \Delta X;$$

and

$$\frac{X_{i+1} - X}{X_{i+1} - X_i} = [X_{i+1} - X_i] - [X - X_i] = 1 - \Delta X.$$

Equation (2) becomes

$$f(X) = f(X_i)[1 - \Delta X] + f(X_{i+1})[\Delta X] \quad (3)$$

The same principle can be applied to obtain the interpolation algorithm for a function of two variables. From Figure 4, $f(X, Y)$ is determined from the magnitude of f_{21} and f_{12} which are in turn determined by $f(X_i, Y_i)$, $f(X_{i+1}, Y_i)$, $f(X_i, Y_{i+1})$ and $f(X_{i+1}, Y_{i+1})$. The magnitudes

$$f_{21} = f(X_i, Y_i)[1 - \Delta X] + f(X_{i+1}, Y_i)[\Delta X]$$

$$f_{12} = f(X_i, Y_{i+1})[1 - \Delta X] + f(X_{i+1}, Y_{i+1})[\Delta X]$$

$$f(X, Y) = f_{21}[1 - \Delta Y] + f_{12}[\Delta Y]$$

Substituting for f_{12} and f_{21}

$$\begin{aligned} f(X, Y) = & f(X_i, Y_i)[1 - \Delta X][1 - \Delta Y] \\ & + f(X_{i+1}, Y_i)[\Delta X][1 - \Delta Y] \\ & + f(X_i, Y_{i+1})[1 - \Delta X][\Delta Y] \\ & + f(X_{i+1}, Y_{i+1})[\Delta X][\Delta Y] \quad (4) \end{aligned}$$

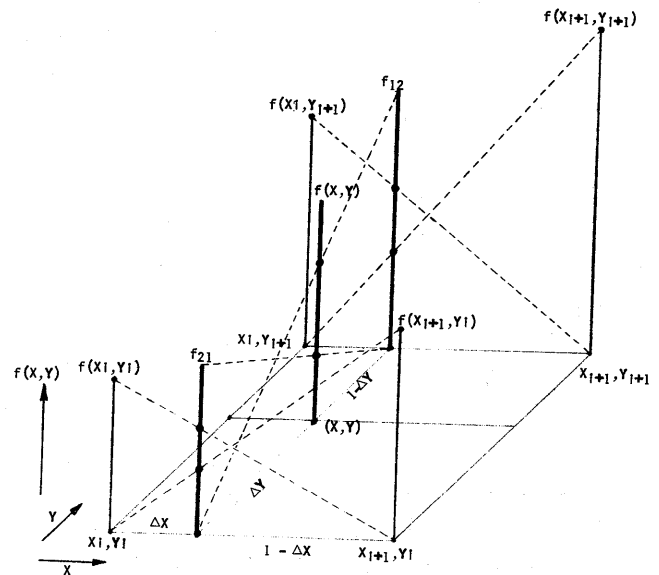


Figure 4—Generation of a function of two variables by superposition

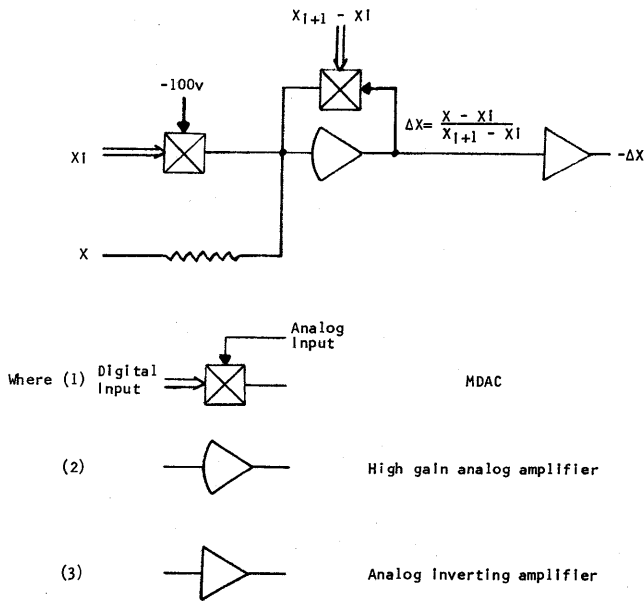
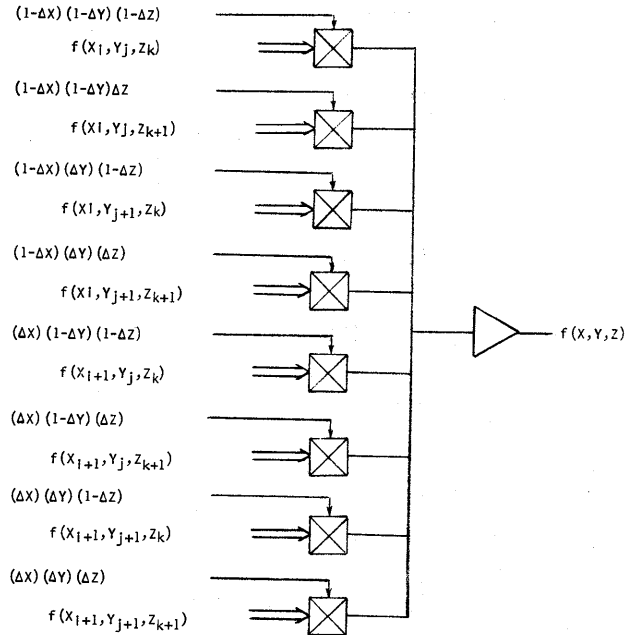


Figure 5A—A typical normalization circuit



- Where (1) Multiplying Digital to Analog Converter
 (2) $f(X_I, Y_j, Z_k), \dots, f(X_{I+1}, Y_{j+1}, Z_{k+1})$ are the functional values at the breakpoints (Digital)
 (3) $(1-\Delta X)(1-\Delta Y)(1-\Delta Z), \dots, \Delta X \Delta Y \Delta Z$ are the weighting coefficients (Analog voltages)

Figure 5C—Implementation of a function of three variables

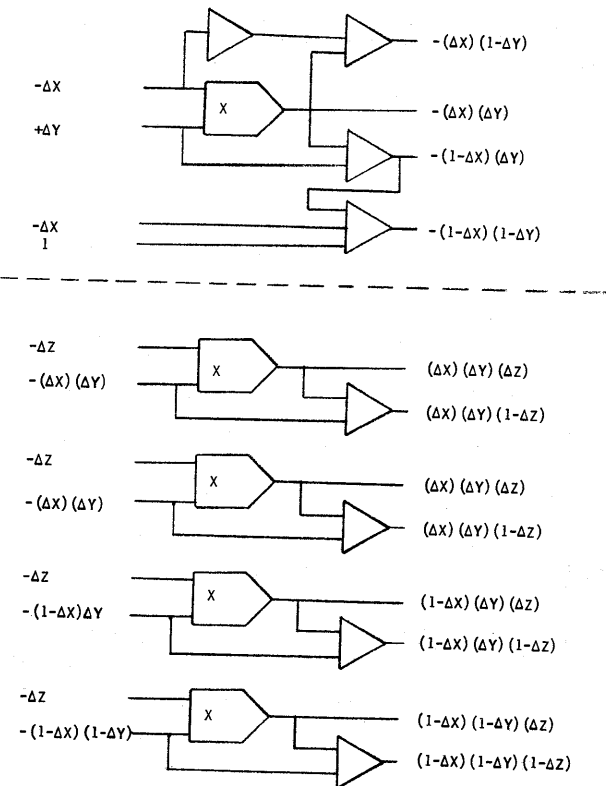


Figure 5B—Generation of weighting coefficients for a function of three variables

The same procedure can be applied to derive the interpolation algorithm for a function of three variables. But we also have observed from the physical picture that the value of the function $f(X, Y)$ at any point (X, Y) is the sum of the contributions of the data points immediately surrounding the point. Furthermore the magnitude of each contribution is proportional to the magnitude of the data point multiplied by the normalized distance from the point (X, Y) to the reference point. The normalized distances are actually weighting coefficients of the data points since their expanded sum is equal to unity.

With the experience gained from the procedure of deriving the interpolation algorithm for a function of two variables we can systematically write down the interpolation algorithm for a function of n variables. As an example, for a function of four variables there are $2^n (n=4)$ data points surrounding any point of interest. Therefore, there will be $2^4=16$ terms contributing to the value of the function $f(X, Y, Z, W)$. Each term will be the value of a data point multiplied by a weighting

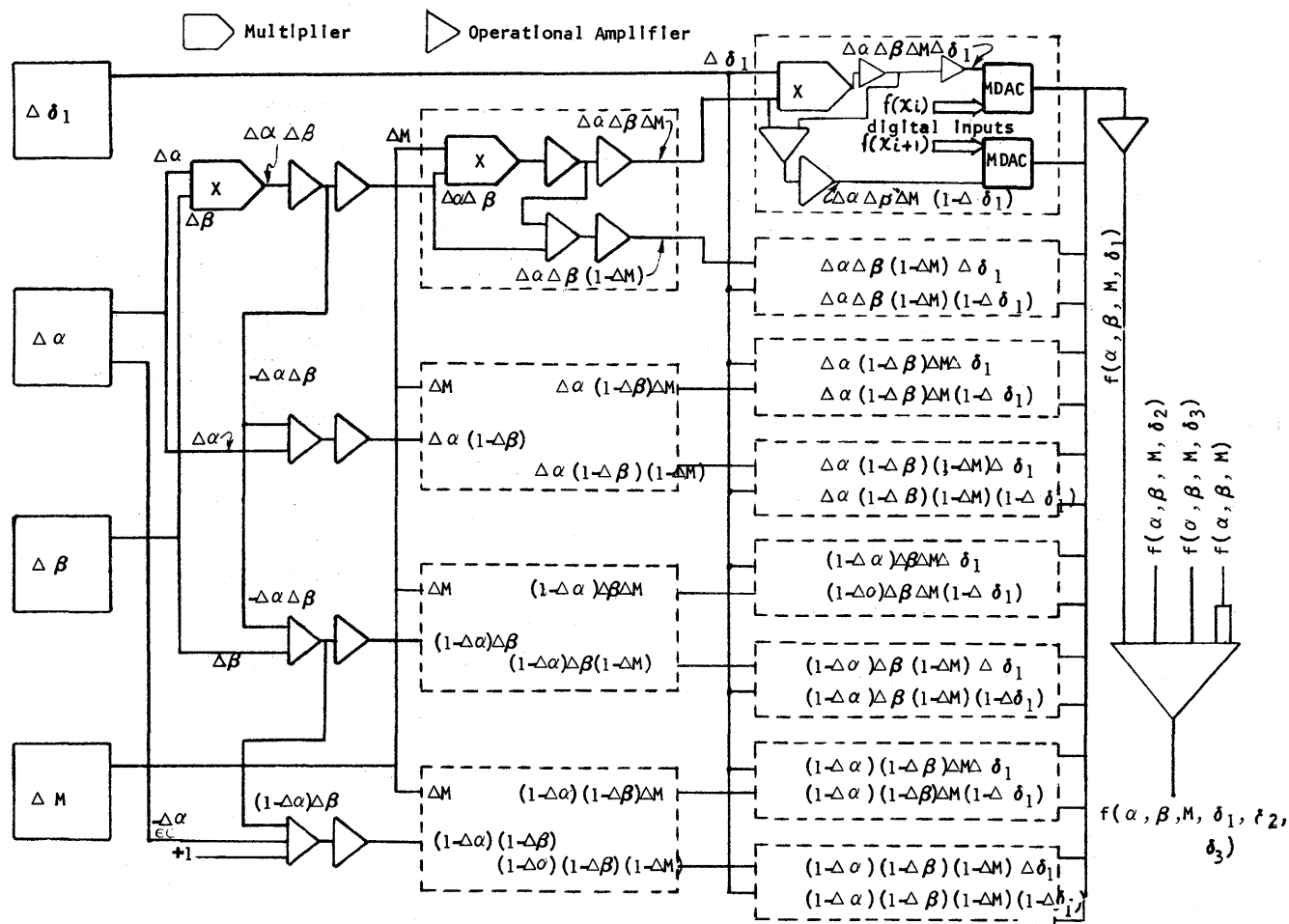


Figure 6—Implementation for the generation of a function of six variables

coefficient or the normalized distance between the reference point and the point of interest.

Thus

$$\begin{aligned}
 f(X, Y, Z, W) &= f(X_i, Y_i, Z_i, W_i)(1-\Delta X)(1-\Delta Y)(1-\Delta Z)(1-\Delta W) \\
 &+ f(X_{i+1}, Y_i, Z_i, W_i)(\Delta X)(1-\Delta Y)(1-\Delta Z)(1-\Delta W) \\
 &+ f(X_i, Y_{i+1}, Z_i, W_i)(1-\Delta X)(\Delta Y)(1-\Delta Z)(1-\Delta W) \\
 &+ f(X_{i+1}, Y_{i+1}, Z_i, W_i)(\Delta X)(\Delta Y)(1-\Delta Z)(1-\Delta W) \\
 &+ f(X_i, Y_i, Z_{i+1}, W_i)(1-\Delta X)(1-\Delta Y)(\Delta Z)(1-\Delta W) \\
 &+ f(X_{i+1}, Y_i, Z_{i+1}, W_i)(\Delta X)(1-\Delta Y)(\Delta Z)(1-\Delta W) \\
 &+ f(X_i, Y_{i+1}, Z_{i+1}, W_i)(1-\Delta X)(\Delta Y)(\Delta Z)(1-\Delta W) \\
 &+ f(X_{i+1}, Y_{i+1}, Z_{i+1}, W_i)(\Delta X)(\Delta Y)(\Delta Z)(1-\Delta W) \\
 &+ f(X_i, Y_i, Z_i, W_{i+1})(1-\Delta X)(1-\Delta Y)(1-\Delta Z)(\Delta W) \\
 &+ f(X_{i+1}, Y_i, Z_i, W_{i+1})(\Delta X)(1-\Delta Y)(1-\Delta Z)(\Delta W)
 \end{aligned}$$

$$\begin{aligned}
 &+ f(X_i, Y_{i+1}, Z_i, W_{i+1})(1-\Delta X)(\Delta Y)(1-\Delta Z)(\Delta W) \\
 &+ f(X_{i+1}, Y_{i+1}, Z_i, W_{i+1})(\Delta X)(\Delta Y)(1-\Delta Z)(\Delta W) \\
 &+ f(X_i, Y_i, Z_{i+1}, W_{i+1})(1-\Delta X)(1-\Delta Y)(\Delta Z)(\Delta W) \\
 &+ f(X_{i+1}, Y_i, Z_{i+1}, W_{i+1})(\Delta X)(1-\Delta Y)(\Delta Z)(\Delta W) \\
 &+ f(X_i, Y_{i+1}, Z_{i+1}, W_{i+1})(1-\Delta X)(\Delta Y)(\Delta Z)(\Delta W) \\
 &+ f(X_{i+1}, Y_{i+1}, Z_{i+1}, W_{i+1})(\Delta X)(\Delta Y)(\Delta Z)(\Delta W)
 \end{aligned}$$

SYSTEM IMPLEMENTATION

In the interpolation algorithm the independent variables X, Y, etc., and their normalized values ΔX, ΔY, etc., are analog signals; the breakpoints X_i, Y_i, etc., and the function data points f(X_i . . .), etc., are digital values. Each function is the sum of several products. Each product is the multiplication of a digital value, f(X_i . . .) by an analog signal, the product of the Δ's, and (1-Δ)'s. This form of the algorithm is

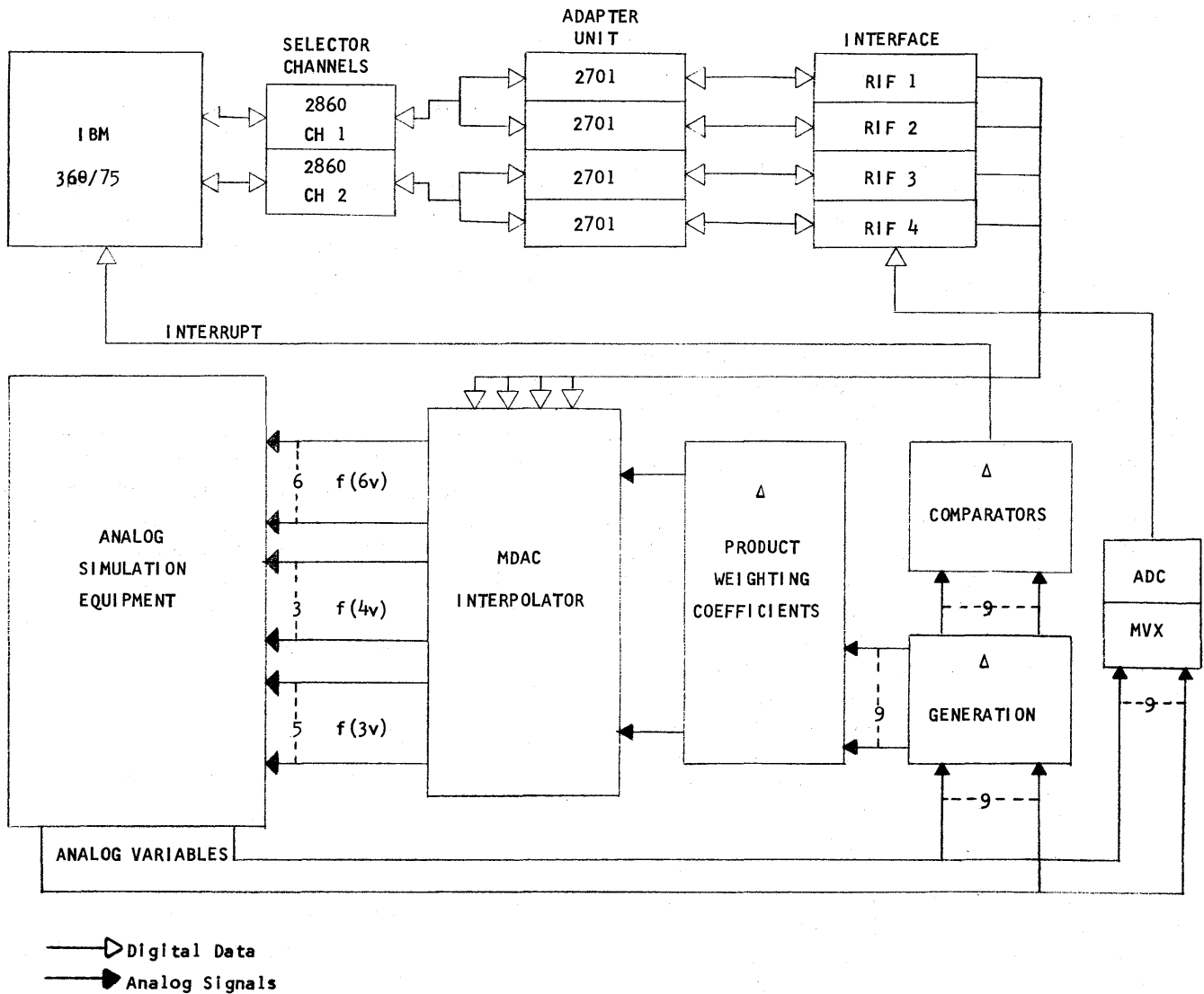


Figure 7—HFSGS implementation

ideally suited for implementation using hybrid computing circuits. The basic computing elements are (1) the multiplying digital to analog converter (MDAC), (2) the analog multiplier, and (3) the operational amplifier. The implementation of a typical normalization circuit is shown in Figure 5A. Figure 5B shows how the products of the Δ 's and $(1-\Delta)$'s can be formed systematically for functions of two and three variables. Figure 5C shows the concise implementation of a function of three variables using MDAC's. Figure 6 shows the implementation of equation (1), a function of six variables.

In a typical flight simulation all aerodynamic coefficients are functions of the same independent variables. Under such conditions one set of normaliza-

tion circuits is sufficient for the generation of all functions of the same variables. This simplifies the total system implementation requirement considerably.

The HFSGS is implemented with an IBM 360/75 digital computer interfaced to four AD/4 analog computers that house hybrid computing elements. A block diagram showing the interconnections of the major subsystems of the HFSGS is in Figure 7. Two-way communication is provided between the digital computer and the interpolation circuits. The DACs and the MDACs in the normalization circuits receive break point values and break point spacing information respectively from the digital computer. The remaining MDACs receive functional values of corresponding break points from the digital computer. The dual data

path provides an effective transmission rate of one million words per second. An analog-to-digital converter (ADC) supplies the values of the analog independent variables to the digital computer. Analog comparators are used for monitoring the outputs of the normalization circuits. The outputs of the comparators are connected through an OR circuit to an interrupt line in the digital computer. Servicing of the interrupt is controlled by the function generation program.

SYSTEM OPERATION

The interpolation circuits for generating the multi-variable functions are programmed on the analog patchboards. The digital computer stores in memory and bulk storage the digitally recorded data representing the functions to be generated. During a simulation, the analog computing elements monitor the independent variables and generate interrupt signals to the digital computer when the value of any variable crosses a breakpoint. This condition occurs when the normalized value of any independent variable goes below zero or above unity. The ADC under the control of the digital computer converts and supplies the values of the independent variables to the digital computer. Based on these values the digital computer calculates data addresses, retrieves, orders and outputs the necessary data to the MDACs. If necessary, it brings in a new page of data from the drum concurrently with the other operations. The computing elements in the analog consoles perform linear interpolation simultaneously and continuously based on the instantaneous value of the independent variables. Negligible phase delay is introduced as the functions are generated. As the value of any variable crosses a breakpoint, the digital computer again updates the MDACs dynamically within 2.5 msec. with new data.

During the 2.5 msec. after an independent variable crosses into a new region and before the MDACs can be updated with the new data, the process of interpolation temporarily becomes extrapolation since the independent variable is outside the defined region. The error of extrapolation depends on the rate of change of data values between adjacent data points. This amplitude error appears as high frequency noise in the simulation.

HFGS CAPACITY

Equipment capable of generating functions of six variables is obviously good for functions of 5, 4, 3, 2 or 1 variables. Table I shows the maximum capacity of the Boeing HFGS. The number of functions

TABLE I—Capacity of the HFGS

No. of functions	212	106	53	26	13	6
No. of Variables	1	2	3	4	5	6

that can be generated increases as the number of variables decreases. A mix in the number of functions and the number of variables is also possible. The total maximum number of independent variables at present is limited to 9.

SYSTEM ACCURACY

One interesting point worth mentioning is the determination of the dynamic accuracy of the HFGS. If a standard sine wave of 100 HZ at 100 volts peak is used to represent the independent variables, the product term of the normalized variables in the interpolation algorithm can become very complex. For instance, let

$$\Delta X = \Delta Y = \Delta Z = \Delta W = \sin wt,$$

then

$$(\Delta X)(\Delta Y)(\Delta Z)(\Delta W) = \sin^4 wt = \frac{3}{8} - \frac{1}{2} \cos 2wt + \frac{1}{8} \cos 4wt$$

The product terms of the weighting coefficients all

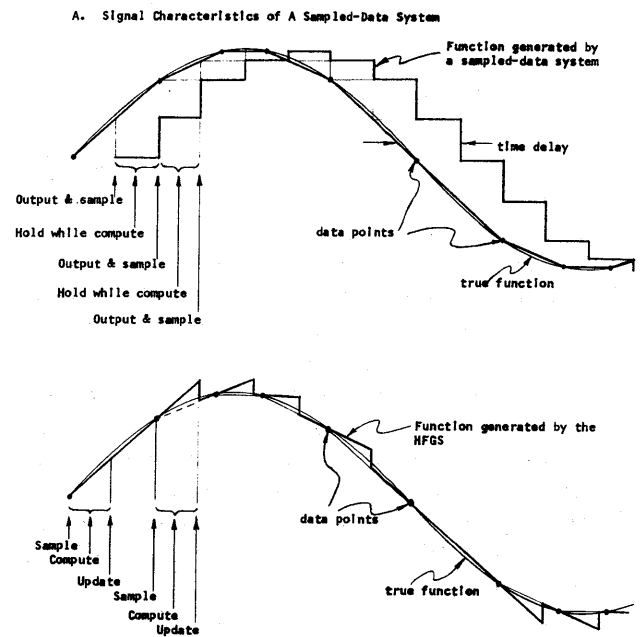


Figure 8—Comparison of output signals

contain harmonics of the original sine wave. The output of the function, in general, will not be a simple sine wave. However, with the understanding of the characteristics of the weighting coefficients, the equation can be simplified for test purposes. The equation for linear interpolation can be rearranged such that all terms containing one normalized variable, such as ΔW are collected in one group and the remaining terms containing $(1-\Delta W)$ in another group. Within each group the common terms ΔW , and $(1-\Delta W)$ are factored out giving:

$$\begin{aligned}
 f(X, Y, Z, W) &= (1-\Delta W)[f(X_i, Y_i, Z_i, W_i)(1-\Delta X)(1-\Delta Y)(1-\Delta Z) \\
 &+ f(X_{i+1}, Y_i, Z_i, W_i)(\Delta X)(1-\Delta Y)(1-\Delta Z) \\
 &+ f(X_i, Y_{i+1}, Z_i, W_i)(1-\Delta X)(\Delta Y)(1-\Delta Z) \\
 &+ f(X_{i+1}, Y_{i+1}, Z_i, W_i)(\Delta X)(\Delta Y)(1-\Delta Z) \\
 &+ f(X_i, Y_i, Z_{i+1}, W_i)(1-\Delta X)(1-\Delta Y)(\Delta Z) \\
 &+ f(X_{i+1}, Y_i, Z_{i+1}, W_i)(\Delta X)(1-\Delta Y)(\Delta Z) \\
 &+ f(X_i, Y_{i+1}, Z_{i+1}, W_i)(1-\Delta X)(\Delta Y)(\Delta Z) \\
 &+ f(X_{i+1}, Y_{i+1}, Z_{i+1}, W_i)(\Delta X)(\Delta Y)(\Delta Z)] \\
 &+ (\Delta W)[f(X_i, Y_i, Z_i, W_{i+1})(1-\Delta X)(1-\Delta Y)(1-\Delta Z) \\
 &+ f(X_{i+1}, Y_i, Z_i, W_{i+1})(\Delta X)(1-\Delta Y)(1-\Delta Z) \\
 &+ f(X_i, Y_{i+1}, Z_i, W_{i+1})(1-\Delta X)(\Delta Y)(1-\Delta Z) \\
 &+ f(X_{i+1}, Y_{i+1}, Z_i, W_{i+1})(\Delta X)(\Delta Y)(1-\Delta Z) \\
 &+ f(X_i, Y_i, Z_{i+1}, W_{i+1})(1-\Delta X)(1-\Delta Y)(\Delta Z) \\
 &+ f(X_{i+1}, Y_i, Z_{i+1}, W_{i+1})(\Delta X)(1-\Delta Y)(\Delta Z) \\
 &+ f(X_i, Y_{i+1}, Z_{i+1}, W_{i+1})(1-\Delta X)(\Delta Y)(\Delta Z) \\
 &+ f(X_{i+1}, Y_{i+1}, Z_{i+1}, W_{i+1})(\Delta X)(\Delta Y)(\Delta Z)] \quad (6)
 \end{aligned}$$

The remaining coefficients inside the brackets are exactly the weighting coefficients of a function with one less variable. If we set the digital values of the data points in the two groups to E_1 and E_2 respectively, and factor them out of each group, we have

$$\begin{aligned}
 f(X, Y, Z, W) &= (1-\Delta W)E_1[(1-\Delta X)(1-\Delta Y)(1-\Delta Z) \\
 &+ \dots + (\Delta X)(\Delta Y)(\Delta Z)] \\
 &+ (\Delta W)E_2[(1-\Delta X)(1-\Delta Y)(1-\Delta Z) \\
 &+ \dots + (\Delta X)(\Delta Y)(\Delta Z)]
 \end{aligned}$$

Since the weighting coefficients inside the brackets

reduce to unity, the equation becomes

$$f(X, Y, Z, W) = (1-\Delta W)E_1[1] + (\Delta W)E_2[1] \quad (7)$$

Equation (7) indicates that whatever signals are used for ΔX , ΔY , and ΔZ , they should sum up to unity for properly chosen values of data points. Equation (7) shows the simple relationship between the output of the function and the input signal ΔW . The use of (7) enables the measurement of small dynamic errors by a simple comparison of output to input while all function generation circuits are being exercised.

The maximum total dynamic error including phase shift for a function of six variables measured at an output signal of $100 \sin 2\pi(100)t$ volts is 1 percent. The measured error is very close to the calculated error based on the individual component specifications.

A COMPARISON WITH OTHER METHODS

Before the hybrid equipment was available for function generation, the aerodynamic coefficients were simulated in an analog aero model. An example of the interpolation polynomial for the coefficient C_L is given below. The mechanization of this polynomial on an analog computer introduces very little phase delay at the output. But, the basic drawbacks are:

1. Many man-months of effort are spent in the derivation of the interpolation polynomial from wind tunnel data.
2. The equation does not produce correct outputs throughout the entire range of interest.
3. The many multiplications and gains in a chain produce unacceptable noise amplitudes.
4. The method requires a major effort to update when new data are obtained from wind tunnel tests.

A sampled-data hybrid system employing a large scale digital computer with simple zero-order hold reconstruction was tested for function generation by the sample, compute, output and hold scheme. The complete cycle using digital interpolation is about 7 milliseconds. The output waveform of this sampled-data system is compared to the waveform of the HFCS in Figure 8. The significant difference is that the HFCS introduces negligible phase delay at the output. With the phase delay created by the sampled-data system the simulation was forced to run at 10 times slower than real-time. Even on a 10 times slower time scale, the method introduces into the simulation a loss in gain margin twice as much as the HFCS running in real-time.

A Sample Equation For An Aerodynamic Coefficient C_l

$$C_l = C_{l\beta}\beta + C_{l\beta^2}|\beta|\beta + C_{l\delta_1}\delta_1 + C_{l\delta_1^2}|\delta_1|\delta_1 + C_{lP}P - C_{Y\Delta Z}CG$$

$$+ C_{l\delta_2}\delta_2 + C_{l\delta_2^2}|\delta_2|\delta_2 + C_{l\delta_3}\delta_3 + C_{l\delta_3^2}|\delta_3|\delta_3 + C_{l\beta^3}\beta^3$$

$$+ [C_{l\alpha\beta}C_{l\alpha\beta^3}\beta^3\alpha + C_{l\alpha^2\beta}\beta + C_{l\alpha^3\beta^3}\beta^3]\alpha^3 \quad \left. \right\} \text{Derivatives depend on the sign of } \alpha$$

$$+ [C_{l\beta\delta_1}|\delta_1| + C_{l\beta\delta_1^2}\delta_1^2 + C_{l\beta\delta_1^4}\delta_1^4]\beta$$

$$+ [C_{l\beta^2\delta_1}\delta_1 + C_{l\beta^2\delta_1^2}|\delta_1|\delta_1 + C_{l\beta^2\delta_1^4}|\delta_1|\delta_1^3]\beta^2 \quad \left. \right\} \text{Derivatives depend on the sign of } \beta\delta_1$$

$$+ [C_{l\alpha\delta_1}\delta_1 + C_{l\alpha\delta_1^2}|\delta_1|\delta_1 + C_{l\alpha\delta_1^3}\delta_1^3]\alpha$$

$$+ [C_{l\alpha^2\delta_1}\delta_1 + C_{l\alpha^2\delta_1^2}|\delta_1|\delta_1 + C_{l\alpha^2\delta_1^3}\delta_1^3]\alpha^2 \quad \left. \right\} \text{Derivatives depend on the sign of } \alpha$$

$$+ [C_{l\alpha\delta_2}\delta_2 + C_{l\alpha\delta_2^2}\delta_2^2 + C_{l\alpha\delta_2^3}\delta_2^3]\alpha$$

$$+ [C_{l\alpha^2\delta_2}\delta_2 + C_{l\alpha^2\delta_2^2}\delta_2^2 + C_{l\alpha^2\delta_2^3}\delta_2^3]\alpha^2$$

$$+ [C_{l\alpha\delta_3}\delta_3 + C_{l\alpha\delta_3^2}\delta_3^2 + C_{l\alpha\delta_3^3}\delta_3^3]\alpha$$

$$+ [C_{l\alpha^2\delta_3}\delta_3 + C_{l\alpha^2\delta_3^2}\delta_3^2 + C_{l\alpha^2\delta_3^3}\delta_3^3]\alpha^2 \quad \left. \right\} \text{Derivatives are zero for negative } \alpha, \text{ and depend on the}$$

$$\text{sign of } \delta \text{ for positive } \alpha$$

$$+ [C_{l\beta\delta_2}\delta_2 + C_{l\beta\delta_2^2}\delta_2^2 + C_{l\beta\delta_2^4}\delta_2^4]\beta$$

$$+ [C_{l\beta^2\delta_2}\delta_2 + C_{l\beta^2\delta_2^2}\delta_2^2 + C_{l\beta^2\delta_2^4}\delta_2^4]\beta^2$$

$$+ [C_{l\beta\delta_3}\delta_3 + C_{l\beta\delta_3^2}\delta_3^2 + C_{l\beta\delta_3^4}\delta_3^4]\beta$$

$$+ [C_{l\beta^2\delta_3}\delta_3 + C_{l\beta^2\delta_3^2}\delta_3^2 + C_{l\beta^2\delta_3^4}\delta_3^4]\beta^2 \quad \left. \right\} \text{Derivatives depend on the sign of } \beta \text{ and the sign of } \delta$$

SUMMARY OF SIGNIFICANT IMPROVEMENTS USING THE HFSGS

The use of the HFSGS has enabled the refinement of real-time flight simulations of high performance systems to a degree never before achieved at Boeing. Some important benefits resulting from the application of the HFSGS are summarized below.

1. The use of recorded data eliminates the tedious process of curve fitting which often fails because the functions are not analytic.
2. The flexibility of a digital computer allows fine adjustments of the aeromodel unattainable by analog computing methods.
3. The parallel, continuous outputs of the HFSGS eliminate the phase delay of sampled-data systems. The illustration in Figure 8 shows that the delay in updating the function values in the MDAC's creates amplitude errors (due to extrapolation) rather than a phase shift. The superior signal characteristics of the HFSGS have enabled the simulation to run in real time with a high degree of confidence in the accuracy of the simulation results. The increase in

simulation speed by a factor of ten over the sampled-data system represents a significant improvement in computing efficiency.

The ease of function changes with the HFSGS have enabled refinement of the aeromodel until simulation data and missile flight test telemetry data closely matched. Observed anomalies were exactly reproduced by the simulation during the post-flight analysis phase. Based on these factors this hybrid technique of multi-variable function generation is considered to be a success.

REFERENCES

- 1 R W HAMMING
Numerical methods for scientists and engineers
1962
- 2 J A PUSTAVER JR.
A multivariable interpolation formula
Air Force Cambridge Research Laboratories Physical Sciences Research Papers No 358 May 1968
- 3 A I RUBEN
Hybrid techniques for generation of arbitrary functions
SIMULATION Volume 7 number 6 December 1966

Problems in, and a pragmatic approach to, programming language measurement

by JEAN E. SAMMET

IBM Corporation
Cambridge, Massachusetts

INTRODUCTION

Although considerable attention has been given to the measurement of *compilers* (e.g., size of compiler, amount of memory needed for compilation, speed of compilation and object program, size of object code), virtually nothing has been done about measurement of *languages*. This is not an empty issue, because there are a number of relevant and significant questions pertaining to programming languages for which we would like to have (quantitative) answers. For example, given three languages which two are most alike? By what criteria? Which of them is most like some fourth language? How could we develop a general ranking or hierarchy for a set of languages according to features so as to handle subsets and extensions? Probably the most important practical question is "For a given application or set of applications, which language is best?"

Among the most frequently used phrases involving languages are the ones indicating one language is a "dialect" of another, or one language is "like" another, i.e., an "L-like language." There is no concrete meaning for these terms. Furthermore, given two "dialects," how do we determine which is closer to the base language?

Finally as an illustration of a different kind of question, consider the problem of having N syntactic forms for accomplishing the same specific task; we need some specific numerical method of comparing them. For example, suppose one language has a key word XYZ, and one "dialect" uses XYZABC while another "dialect" uses RST; which is closer to the original?

The entire field of programming languages is quite subjective; opinions are used more often than facts. Part of the reason for this is the lack of numerical values which can be associated with languages, and hence the lack of concrete data. Development of methods for quantification should help improve objec-

tivity. Eventually such measurements should help improve the selection of a language, the design of new languages, the modification of existing ones, and even in implementation.

The second section establishes the problem by discussing currently used terms, the practical need for comparison, types of measurements, and elements not included in the current approach to the problem. The third and fourth sections discuss approaches to measurement of non-syntactic and syntactic characteristics, respectively, including relevant features, the numerical approach, and examples. A brief summary is given at the end.

The only related published work seems to be that of Goodenough.¹ His paper concentrates on syntax and semantics from a linguistic (not a numerical) viewpoint. This paper takes a very pragmatic approach and emphasizes many of the intangible aspects of languages.

ESTABLISHMENT OF MEASUREMENT PROBLEM

Currently used terms

There are a number of terms which are currently used in discussing programming languages which imply some type of measurement. Unfortunately this measurement is very likely to be subjective. As the simplest illustration, consider the term "dialect" which is generally used for a language purportedly very similar to some other language; very often a dialect is merely a particular implementation of a well known language with some "trivial" changes made. Well defined dialects usually arise by making "minor" syntactic changes, e.g., restricting the number of characters in a data name, eliminating certain options in a particular command, and/or adding some particular feature or removing a

restriction which is in the original language. As indicated in the previous section, one of the problems that constantly faces us is the situation in which we have two dialects of a given language and no way of measuring them relative to each other or to the base language.

Another popular term which is frequently heard is the phrase "L-like language." This usually refers to a language which is similar in spirit and notation to language L, but differs from it markedly enough not to be considered merely a dialect. Thus we hear of "ALGOL-like" languages, or "PL/I-like" languages (e.g., REDUCE and MAD/I respectively). Not only do we have the same problem of measuring the "likeness" that exists with dialects, but in addition we have no way of indicating when a language stops being a dialect and when it starts becoming an L-like language.

In both these cases the primary issue is one of syntax; semantics generally plays only a minor role.

This paper does *not* provide firm definitions of the terms "dialect" and "L-like language"; however, the types of measurements discussed do provide an approach which will help in defining these terms. As a first approximation, we might (arbitrarily) say that a language which has a syntactic deviation of 20 percent from another language is a dialect, whereas a deviation between 20 percent and 50 percent would be considered "language-L like." Beyond 50 percent it might be truly considered a different language.

The questions of subsets and extensions are somewhat more easily dealt with. This author has already stated in Reference 3 the following definition of subset: A language S is considered a proper subset of a language L if (1) there are some programs which can be legally written in L which cannot be legally written in S; (2) all legal S programs are legal L programs; and (3) the results from a program written in S when executed with an S compiler are the same as the results obtained from an L compiler on the same machine, except for those aspects which are implementation dependent.

From that definition of subset, we can then easily say that a language E is an extension of a language L if L is a subset of E.

If we look at only subsets or extensions which are arranged in a hierarchical fashion then there is very little problem. Thus we could have a language L with extensions $E(1)$, $E(2)$, ..., $E(n)$ where $E(i)$ is a proper subset of $E(i+1)$ for all i . A similar concept can apply to subsets. However, in actual practice the situation is seldom that simple and what happens far more frequently is that there are two languages, both of which are extensions (or subsets) of the same base language but neither of which is properly contained in the other. We need to have some way of measuring the size of the extension (or subset) when there are non-nested

extensions (or subsets). For example, if one extension of a language contains a new data type, and another one contains a new command, which is really a "larger" extension of the base language? Similarly if one subset removes an input/output command and another one eliminates a double precision facility, which is the smaller subset?

Finally, as the worst situation we very frequently have what can be called an "L-like extended subset." This is a situation in which a language L has a subset S with some minor deviations (called S') but some features are added to the subset which are not in the language L (say S'+). The result is surely an L-like language (or might even be considered merely a dialect) but we cannot say anything more quantitative than that. If we have two such situations (say $S_1'^+$ and $S_2'^+$) we have no way of measuring the amount of differences involved. A good example of this is CPS and RUSH, both of which are PL/I-like extended subsets.

Practical need for comparison

There are at least two broad classes of people who are concerned in a very practical way with these measurements and comparisons, completely aside from any theoretical interest. One is the user and the other is the implementor.

The user in general is concerned with the problems of relevance and compatibility. In the case of relevance, he is very concerned with the usefulness of a particular language for a particular problem or a broad application area. However, any resulting decisions on what to actually use are generally based on intuition. The user badly needs a way of measuring languages in terms of their relevance to his needs. He is also greatly concerned with all of the issues pertaining to compatibility. For example he would like some way of knowing which of two languages is most like some other language, because use of the "closer one" will ease his training problem and/or improve compatibility and hence ease his potential conversion. On the other hand, given two languages both of which seem to meet his needs it would be very desirable for him to have some way of measuring which of these two was "closer" to his needs. A nonnumerical approach to this for one case is described in Reference 2.

The implementor is slightly less concerned in a practical way and slightly more interested from the theoretical viewpoint. As part of the practical considerations, the implementor is likely to want to measure languages because he may want to consider which techniques that have been previously used in a compiler

for a "similar" language are applicable; if there was some reasonable measure of the deviations of the language he might be able to tell. In another instance, the implementor might have a compiler for a given language and be attempting to determine how wide a deviation in the language could still be handled by the same compiler. (In many cases the measurement may be irrelevant because one language might be much closer to another one numerically but the deviations would cause far more drastic changes in compiling techniques than from a language which had a bigger numerical deviation.)

Types of measurements

There are many types of measurements that can be applied, but not all are meaningful in all cases. One important type of measurement is that of a single language against some fixed numeric scale for a particular characteristic. Thus we might conceivably have an absolute scale for generality which is certainly one characteristic of a language; a numerical value could be given (providing we could establish the scale in the first place which is extremely difficult).

Another very important type of measurement is of a single language with respect to a given application. Thus while we might measure a particular language for generality considered across the scope of all desired computations, in more realistic circumstances we would be likely to consider a particular language against a specific application or a broad area of applications.

Finally, we are frequently interested in measurements between two languages. This tends to be simpler in many cases because it is easier to indicate that language A is more general than language B without actually worrying about a numeric scale. If we introduce a third language and we wish to rank them, we can still do this on a pair-wise basis; however, it will become increasingly harder to compare more than two languages unless we use some type of numeric scale.

It is important to realize that there is a significant difference between measurements of a *language* and measurements of a *program* written in that language. The former can be developed once, from the given specifications, whereas the latter will literally depend upon who does the programming.

Elements not included in the current approach to the problem

In this first introduction to the overall problem there are several elements which will *not* be considered. First and foremost, no attempt will be made to include any

type of formal semantics. We will be dealing primarily with syntax, and will implicitly use semantics only in an intuitive fashion, i.e., we know intuitively what a particular syntactic construction is meant to do. Furthermore, syntactic measurements will be shown in this paper only through a limited example i.e., specific methods of providing detailed measurements of syntactic elements are not included. Finally, no attempt will be made to establish an absolute scale for individual features.

It should be emphasized that the problem of measuring languages specifically excludes measurement or numerical comparisons of implementations. Thus this aspect does not need to be included in any approach to the problem.

APPROACH TO MEASUREMENTS BASED ON NON-SYNTACTIC CHARACTERISTICS

Relevant features and viewpoints

As the first step in measuring programming languages with respect to their non-syntactic characteristics, it is necessary to list the relevant features or elements or characteristics (e.g., consistency, ease of reading) that are to be measured, where subfeatures will be used as appropriate. Figure 1 provides this list. However, it is not necessarily meaningful to measure all features in all ways. The characteristics of the language can be measured with respect to the following viewpoints:

absolute scale (if one exists)	
user	(U)*
implementor	(I)*
one other language	(OOL)*
two or more other languages	(TML)*
specific application	(SA)*
application area	(AA)*

A particular characteristic may be irrelevant or of minor importance from some viewpoints (e.g., relevance to an application is not significant to the implementor). In a few cases the relevance is questionable.

Numerical approach

To the extent that numbers can be supplied, the following simple techniques will be used. Absolute scales should be established in the range 0 to 1, with 1 representing the maximum of the char-

* Abbreviation used in Figure 1.

<u>Non-Syntactic Characteristics</u>	<u>Viewpoints</u>					
	<u>U</u>	<u>I</u>	<u>OOL*</u>	<u>TML*</u>	<u>SA</u>	<u>AA</u>
CONSISTENCY	s	g	g	g	NA	NA
This deals with internal contradictions in rules, or exceptions to rules. For example a language might say that blanks are irrelevant except in particular cases. This language would then have a certain amount of inconsistency within it.						
EASE OF						
READING	g	NA	g	g	s	s
WRITING	g	NA	g	g	g	g
DEBUGGING (from the language viewpoint only)	g	g	g	g	g	g
MAINTENANCE	?	g	g	g	?	?
LEARNING	g	NA	g	g	g	g
CONVERSION	g	?	g	g	g	g
IMPLEMENTATION	?	g	g	g	?	g
Each of these "ease" features calls for a different measurement since what is easy to read is not necessarily easy to write, etc.						
ENVIRONMENT INDEPENDENCE						
MACHINE INDEPENDENCE	g	g	g	g	g	g
OPERATING SYSTEM INDEPENDENCE	g	g	g	g	g	g
ON-LINE VERSUS BATCH INDEPENDENCE	s	g	g	g	g	g
While all programming languages are fairly machine independent, some have some implicit hardware dependencies which could be simulated but only with great inefficiency (e.g. read tape backward). Furthermore there are cases of language dependencies on the operating system (e.g. handling of STOP command or its equivalent). Some languages can only be implemented effectively in on-line or batch modes.						
GENERALITY	?	s	g	g	s	g
This is essentially equivalent to the phrase "general purpose". A completely general purpose language could be used <u>effectively</u> for all applications and problems. There is no such language today.						
NATURALNESS	g	NA	g	g	g	g
This deals with the intuitive resemblance of the programming language itself to the way in which an individual would describe the problem or give instructions about how to solve it to another person.						
NON-PROCEDURAL	g	s	g	g	s	g
It is this author's contention that non-procedural is a relative term which changes as the state of the art changes. At a given point in time one can talk about the amount of non-procedurality in a particular language.						

Figure 1—Measurement of Non-Syntactic Characteristics

	U	I	Viewpoints		SA	AA
			OOL*	TML*		
RELEVANCE TO APPLICATION AREA This includes the whole gamut of notation, features, etc. which would be used for a particular application area.	g	NA	g	g	g	g
RELEVANCE TO SPECIFIC APPLICATION In contrast with the above, a particular application may impose different demands on the language than a broader area and therefore different measurements will be needed.	g	NA	g	g	g	g
SIMPLICITY This provides some measure of the complexity of the rules in the language but also must bear some relationship to the amount of generality. A very narrow language can be very simple because not many rules are needed; a very general language may need more rules. Thus simplicity really should be measured both in absolute terms and also as a ratio of the generality. To simplify matters only the absolute scale will be considered.	?	?	g	g	s	s
SUCCINCTNESS This contrasts with verbosity, i.e., how many pencil strokes are needed to convey a particular concept.	s	g	g	g	s	s
USE AS A HARDWARE LANGUAGE Some languages are definitely defined using a character set which is not readily available on normal equipment. This affects the direct and immediate use of the language as input to a computer.	g	s	?	?	s	s
USE AS A PUBLICATION LANGUAGE Some languages are particularly well designed for use in normal publication media and this can be used as a measurement.	?	NA	?	?	s	s

Column Headings:

U = User
I = Implementor
OOL = One Other Language

TML = Two or More Other Languages
SA = Specific Application
AA = Application Area

Column Entries:

g = of great importance
s = of small importance

? = relevance is questionable
NA = not applicable

* While these two columns are identical in this formulation, it seems advisable to keep both for potential changes as this table is revised and refined.

Figure 1—(Continued)

TABLE I—Measurement of Languages from Viewpoint of A User Writing A Payroll Program*

Feature	Weighting* Factor	Normalized Weighting Factor	Raw Scores*		Normalized Weighted Scores	
			COBOL	PL/I	COBOL	PL/I
Consistency	NA		—	—		
Ease of						
reading	.9	.09	1	.3	.090	.027
writing	.8	.08	1	.5	.080	.040
debugging	.5	.05	1	.8	.050	.040
maintenance	.2	.02	.7	1	.014	.020
learning	.3	.03	1	.5	.030	.015
conversion	.8	.08	1	.2	.080	.016
implementation	.2	.02	1	.3	.020	.006
Environment independence						
machine independence	.9	.09	.8	1	.072	.090
operating system independence	.9	.09	1	.4	.090	.036
on-line vs. batch independence	.1	.01	1	1	.010	.010
Generality	.1	.01	.2	1	.002	.010
Naturalness	.7	.07	1	.2	.070	.014
Non-procedural	.5	.05	1	1	.050	.050
Relevance to application area	NA		—	—		
Relevance to specific application	1	.10	1	.6	.100	.060
Simplicity	.5	.05	1	1	.050	.050
Succinctness	.5	.05	.3	1	.015	.050
Use as hardware language	1	.10	1	1	.100	.100
Use as publication language	.1	.01	1	1	.010	.010
Totals	10.0	1.00			.933	.644

* These values are based on the author's personal judgment and are somewhat arbitrary. The numbers are meant primarily for illustrative purposes.

acteristic. It will be assumed that the relationships are linear, i.e., if one language has a measurement of .9 on an absolute scale (with the maximum of 1) and another one has a measurement of .3 it will be assumed that the first had three times more of the characteristic than the second. Since the types of elements being measured are not in the same units, each measurement must be normalized. This is accomplished by doing the following: Where comparisons of two or more languages are made, the one having the highest value will be assigned the value 1 and all others assigned the appropriate ratio value. Using this technique also permits us to obtain quantitative results without assigning an absolute measurement to any language elements.

A fundamental assumption in attempting to measure programming languages from any point of view is that the criteria to be used will vary depending on both the individual and the viewpoint from which he makes the measurement. What is important to one person is unimportant to another; what is vital in one type of measurement is insignificant in another. For example, from the viewpoint of relevance to an application area the actual syntactic form of a loop control statement

may not make much difference, whereas rules on formation of data names may be very significant. The views of two individuals will differ even if they make the same comparisons from the same viewpoints (e.g., use, implementation). In order to accommodate these individual differences, each person attempting to do a measurement will assign his own weighting factors *each time* he makes a measurement. These will be normalized so that the total is 1 and then a numerical score can be obtained by multiplying each weighting factor against the numerical value (already normalized) for the related characteristic and adding them. This then produces a number—albeit a very crude one—which represents a measurement of a particular language *using the viewpoint* (i.e., the criteria) *of the person making the measurement*.

TABLE II—Types of Changes to be Made to a Base Language

- None (i.e., same as base language)
- Deletion (i.e., subset)
- Addition (i.e., extension)
- Substitution (of one word or character for another)
- Optional (in new version instead of required in base)
- Required (in new version instead of optional in base)
- Other change (i.e., not shown above)

TABLE III—Features in Base Language and Two Other Versions*

Feature	Base Language	Version 1	Version 2
Character Set	A-Z = < > . , ** + - * /	As shown, plus use of 7 for NOT	Uses ↑ instead of **
Literals	Single quotes required	Single or double quotes required	Double quotes required
Multiple Assignment Statements	Not allowed	Allowed	—
Keyword LET before assignment statements	Optional	Required	Optional use of key words COMPUTE or LET
Built in functions	11 specific functions	—	11 as specified plus 2 more functions
Computed GOTO DATA statement	Not defined Number or character strings	— Numbers or character strings or expressions	Allowed Numbers or character strings or expressions
MATRIX inversion statement END	Present Required as last statement in program	Not allowed Optional	— —
Statement numbers	Required	—	Optional

* The entry — indicates that the feature has the same specification as the base language.

Specific example

Table I shows the author's evaluation of the non-syntactic features of COBOL and PL/I made from the viewpoint of a user wishing to write a payroll program. Since this author believes intuitively that COBOL is better suited for this application, it is not surprising that the numerical results confirm that; however the figures were not manipulated, i.e., weighting factors and raw scores were assigned without any fudging of the figures except to increase two weighting factors by .1 each to make the total 10 instead of the original 9.8 which had come about naturally. Readers are invited to replicate this experiment for themselves.

APPROACH TO MEASUREMENT OF SYNTAX

Relevant features

In considering the syntactic elements of the language, the following are some of the parameters which must be considered as elements in the measuring system:

- reserved words (existence, number, exact words themselves)
- punctuation, including handling of blanks and literals
- length of user-defined identifiers (i.e., data names, statement names)

- mandatory versus optional usage of words or features
- program structure (e.g., blocks, procedures, sequencing rules)
- data types
- commands
- declarations

There are many different ways of measuring, and not all the items in the above list can be measured the same way. For example, it is easy to compare lists of reserved words, but hard to compare program structures. Then again, which is more important in com-

TABLE IV (a and b)—Scores Assigned to Types of Changes*

Change Type	(a)	(b)
	User Viewpoint of Compatibility of Program in the Base Language to New Language	User Viewpoint of Generality of New Language with Respect to Base Language
None	+1	0
Deletion	-1	-1
Addition	0	+1
Substitution	- .5	0
Optional	- .1	+ .8
Required	- .8	- .5
Other change	- .7	0

* These scores are based on the author's value judgments and are somewhat arbitrary; they are meant primarily for illustrative purposes. The scores are not normalized because that seems to be unnecessary. However, maximum values of +1 and -1 are used.

TABLE V—Measurement of Features From Viewpoint of Compatibility

Feature	Weighting* Factor	Normalized Weighting Factor	Raw Scores**		Normalized Weighted Scores	
			Version 1	Version 2	Version 1	Version 2
Character Set	.8	.13	0	-.5	0	-.068
Literals	.8	.13	0	-.5	0	-.068
Multiple Assignment Statements	.4	.07	0	+1	0	+.07
Keyword LET before assignment statements	.9	.15	-.8	-.7	-.12	-.105
Built in Functions	.6	.10	+1	0	+.10	0
Computed GOTO	.5	.08	+1	0	+.08	0
DATA statement	.8	.13	0	0	0	0
MATRIX inversion statement	.2	.03	-1	+1	-.03	+.03
END	.1	.02	-.1	+1	-.002	+.02
Statement numbers	.9	.15	+1	-.1	+.15	-.015
Totals	6.0	.99			+.178	-.136

* These values are based on the author's personal judgment of the importance of the feature with regard to compatibility and the other parts of the language. The values are meant primarily for illustrative purposes.

** Based on Tables III and IV(a).

Note: The significance of the specific numbers +.178 and -.136 cannot be stated quantitatively. What *can* be concluded is that Version 1 is more compatible with the base language than Version 2.

paring reserved words—length, or similarity of letters? i.e., is the word XYZ closer to the word XYZABC or to the word ABC? The answer again depends on the viewpoint. The primary viewpoints are user and implementor, each of whom may use specific criteria (e.g., compatibility, generality) and/or many of the non-syntactic characteristics shown in Figure 1.

Numerical approach

The approach here is similar to that used for the non-syntactic characteristics. The major types of changes to the syntax of a language are shown in Table II. (Note that the common term "restriction" actually becomes one of the listed items for each specific case.)

TABLE VI—Measurement of Features From Viewpoint of Generality

Feature	Weighting* Factor	Normalized Weighting Factor	Raw Scores**		Normalized Weighted Scores	
			Version 1	Version 2	Version 1	Version 2
Character Set	.5	.10	+1	0	+.10	0
Literals	.5	.10	+1	0	+.10	0
Multiple Assignment Statements	.3	.06	+1	0	+.06	0
Keyword LET before assignment statements	.8	.16	-.5	0	-.08	0
Built in functions	.6	.12	0	+1	0	+.12
Computed GOTO	.5	.10	0	+1	0	+.10
DATA Statement	.7	.14	+1	+1	+.14	+.14
MATRIX inversion statement	.3	.06	-1	0	-.06	0
END	.1	.02	+.8	0	+.016	0
Statement numbers	.7	.14	0	+.8	0	+.112
Totals	5.0	1.00			+.276	+.472

* These values are based on the author's personal judgment of the importance of the feature with regard to generality and the other parts of the language. The values are meant primarily for illustrative purposes.

** Based on Tables III and IV(b).

Note: The significance of the specific numbers +.276 and +.472 cannot be stated quantitatively. What *can* be concluded is that Versions 1 and 2 are both more general than the base language and Version 2 is more general than Version 1.

Depending on the viewpoint from which the measurement is to be made, the individual assigns a value from +1 to -1 with the "best" and "worst" changes at the extremes. A change which is irrelevant to the particular measurement being made is assigned the value 0.

Each syntactic feature under consideration is assigned a weighting factor between 0 and 1 to represent the individual's judgment of its importance from the particular viewpoint involved. These are then normalized.

Raw scores for each syntactic feature are obtained by determining the type of each syntactic deviation and assigning the appropriate "measure of change" score. Multiplication of normalized weighting factors by the raw scores, followed by addition, yields a number representing the syntactic deviation from the base language according to the specified viewpoint.

Specific examples

In order to illustrate the types of measuring on syntax that can be done, some hypothetical cases are taken. In Table III, some syntactic features in a (hypothetical) base language are specified, and then two versions of the base language are defined with respect to those same characteristics. (The language is intuitively BASIC, but that is not significant to the discussion.) Tables IV(a) and IV(b) assign a score to each of the change types, considered from two different points of view—compatibility of a program in the base language to a new language, and generality with respect to the base language. Tables V and VI show the raw and weighted scores for each version, from the viewpoints of compatibility and generality, respectively. The results show (a) Version 1 is more compatible with the base language than Version 2, and (b) Versions 1 and 2 are

more general than the base language, with Version 2 more general than Version 1.

SUMMARY

This paper has attempted to provide an introduction to the need for, and a pragmatic approach to, measuring programming languages. Commonly used terms such as "dialect" were shown to have only an intuitive meaning although numerical measures could and should be defined. Two very simple numerical approaches to obtaining some quantitative results for syntactic and non-syntactic characteristics were outlined. A set of non-syntactic characteristics was described, and the major syntactic parameters involved in measurement were shown. Three specific examples illustrate the techniques involved.

The approach shown here is not yet ready for practical usage except in very simple cases, and the numerical techniques have deliberately been kept simple to make further explorations of this problem easy to do. The actual numbers used were primarily for illustrative purposes and should not be considered as absolute values to be used in all similar cases.

REFERENCES

- 1 J B GOODENOUGH
*The comparison of programming languages:
a linguistic approach*
Proceedings ACM 23rd National Conference 1968
- 2 H HESS C MARTIN
TACPOL—a tactical C & C subset of PL/I
Datamation Vol 16 No 4 April 1970
- 3 J SAMMET
Programming languages: History and fundamentals
Prentice-Hall Englewood Cliffs N J 1969

The ECL programming system*

by BEN WEGBREIT

Harvard University
Cambridge, Massachusetts

INTRODUCTION

ECL is a programming language system currently being implemented as a research project at Harvard University.** Its goal is an environment which will significantly facilitate the production of programs. In this paper, we describe the motivation for this project, present the approach taken in its design, and sketch the resulting ECL system. Detailed treatment of specific aspects of the system are found elsewhere.^{1,2}

Programmers, whether professionals or casual users, manufacture a unique product, programs: objects, often large, which must be coded, modified, debugged, verified, made efficient, and run on data. In providing an environment for this manufacturing, four goals were considered primary:

1. To allow problem-oriented description of algorithm, data, and control over a wide range of application areas.
2. To facilitate program construction and debugging.
3. To allow and assist in the development of highly efficient programs.
4. To facilitate smooth progression between initial program construction and the final realization of an efficient product.

ECL consists of a programming language and a system built around this language to meet these goals.

The language component, called EL1, includes most of the concepts of ALGOL 60, LISP 1.5, and COBOL. It provides standard arithmetic capability on scalars and multidimensional arrays, dynamic storage alloca-

tion with automatic storage reclamation, record handling, and algorithm-independent data description. Further, it provides facilities which allow the programmer to define extensions to the language to tailor it to each particular problem area. New data types, new operators, new syntax and new control structures can be added to the language enabling the program to model directly the objects, unit operations, relations, and control behavior of each problem domain. For example, list processing, matrix arithmetic, string manipulation by pattern matching and replacement, and discrete simulation can all be carried out in EL1 by appropriate extensions.

To aid program construction and debugging, the ECL system has been designed for use in an interactive on-line fashion.† Programs can be composed at the console using a text editor and run interpretively with appropriate levels of error checking, tracing, and conditional suspension. With execution suspended, the programmer can examine data or program, modify either, and resume. Any variable may be declared "sensitive"; changes to its value are monitored and an interrupt generated whenever a programmer-specified predicate associated with the variable becomes true.

Several system facilities contribute to the construction of efficient programs. One is the compiler. Variables can be data typed so that the compiler can perform type checking, compile in type conversion, and choose among alternative procedure bodies on the basis of argument data types. The compiler can be called at any time, so it is possible to write procedures which compile themselves or other procedures. To allow economical use of storage, the language allows packed

* This work was supported in part by the U.S. Air Force, Electronics System Division, under Contract No. F19628-68-C-0101 and by the Advanced Research Projects Agency under Contract No. F19628-68-C-0379.

** The current implementation is on a PDP-10 running under the 10/50 monitor. Versions for other machines are contemplated.

† This is not to the neglect of batch processing. Any interactive language can be used in batch mode if the job control commands that would normally come from the console are taken from a file and results which would normally appear on the console are written to a second file. ECL allows such switching of command streams, so that batch processing falls out as a subcase of its normal mode of operation.

data (e.g., bits, bit strings, bytes, and byte strings) and operations on such data objects. This is carried out in a machine-independent notation and representation so that programs using this are not tied to a particular machine. To allow the construction of efficient programs which include asynchronous components, ECL includes multiprogramming and a programmer-controllable interrupt system.

Efficiency, in any metric, is seldom gained at one fell blow; programs are only relatively stable. Even after code is checked out with the interpreter and compiled, it is usually changed and frequently requires debugging. Further, it is sometimes necessary to compile part of a program in order to get sufficient speed to test an algorithm against a large data base. Since the road is filled with relapses, it is important to allow smooth progression and regression between initial construction and final product. It should scarcely need saying that the languages acceptable to the interpreter and compiler are identical and that compiled and interpreted code may be freely intermixed with no restrictions. For example, the result of compiled code may be used as an argument to interpreted code; a *goto* in interpreted code may lead back into compiled code; variables local to compiled code may be accessed by interpreted code, etc. A less familiar concept, but equally fundamental, is the notion that compilation is not all or nothing. In ECL, compilation can be carried out to any level depending on the amount of information supplied to the compiler: specifically, the number of program components that the programmer is willing to accept as being invariant. The more invariants, the better the compiled code. As with interpreted code, the execution of compiled code may be broken (either by an internal condition or an external interrupt) to allow intervention by the programmer, e.g., for debugging purposes.

The primary motivation for, and the intended use of, the ECL programming system is "difficult" programming efforts. That is, projects which could otherwise be carried out only with considerable waste of human or machine resources. It is our intention that ECL be usable for production programming. Hence the emphasis on machine efficiency. This is not to say that the requirements of interactive usage have been slighted in system design. Quite the contrary, we view good interaction capability and a well-engineered debugging facility as significant tools in tackling a difficult programming project. The utility of on-line debugging should be clear. Equally important is the use of an interactive capability in developing and refining algorithms. Still more important is the use of interaction in allowing measurement of program behavior and the

attendant optimization based on knowledge of this behavior.

SYSTEM ORGANIZATION AND DESIGN PHILOSOPHY

Before discussing ECL in detail, it will be useful to outline its internal organization and discuss the philosophy which underlies its design.

Normally, one uses ECL on-line, communicating with the system via a console. As seen by the programmer, ECL is an executor of input commands. Syntactically, commands correspond roughly to statements of an algebraic language; semantically, commands embrace all actions expressible in the system. Hence, commands include: conventional algebraic statements, definitions used to construct new procedures and operators, and the "job control" statements of a batch processing system such as instructions to compile procedures, transact with data sets, create and destroy processes, etc.

As seen by ECL, the programmer is a source of input commands. We will take the system's point of view. It reads and parses each command, interprets it, and turns to the next command. Since commands include

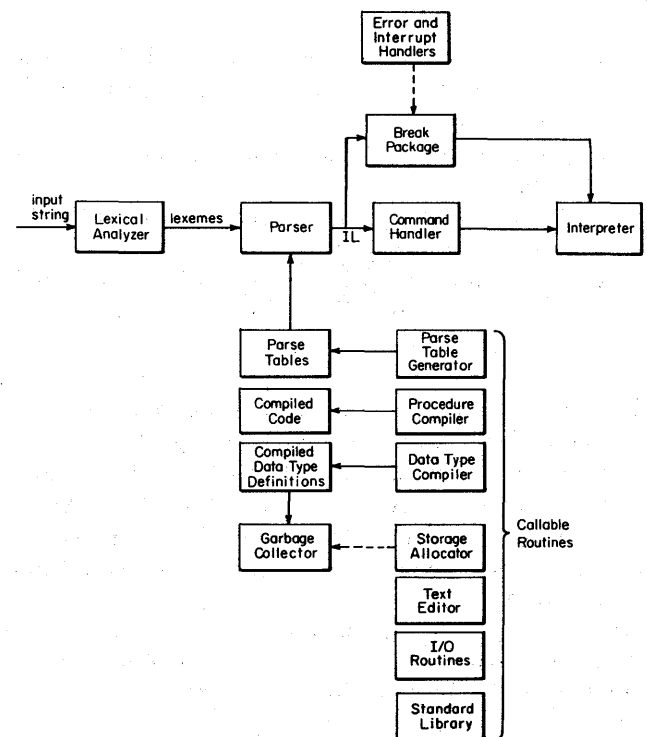


Figure 1—Primary system modules

calls on procedures which may be programmer-defined, the interpretation portion of the cycle may set off the running of a compiled program.

At the heart of ECL is the *command handler*—the routine which controls the above command loop. It has two main components: the *parser* and the *interpreter* (c.f. Figure 1). The parser calls on a *lexical analyzer* to decompose the input stream into lexemes. The parser then analyzes the lexeme stream as directed by parse tables previously derived from a syntactic specification of the language. Both the input source and the parse tables may be changed by commands, so that the source of commands and the language in which commands are expressed are subject to change by the programmer. The output of the parser is a representation of the command as a linked list. Constituent syntactic units are represented by sublists, recursively. The command handler calls on the *interpreter* to execute the command. When this is completed, control returns to the command handler which outputs the result and then calls on the parser for the next command.

The list structured representation has two uses. On the one hand, it can be executed directly by the interpreter; on the other, it is a convenient form of input to the compiler. This achieves several economies. A program need be parsed only once, on input. Hence the interpreter does *not* reparse a line each time it is encountered during execution, e.g., in a loop. Also, the compiler is considerably simplified since it is not at all concerned with parsing.

Most commands will be function calls, i.e., the application of a routine (procedure or operator) to a set of arguments. Routines initially available in ECL include:

1. The conventional arithmetic, relational, and trigonometric routines.
2. A set of I/O routines.
3. A routine for defining new procedures and operators.
4. The compiler.
5. Routines to define new data types.
6. Routines to change the parse tables, thereby changing the syntax of the language.
7. Routines to allocate storage, and a garbage collector to reclaim storage no longer in use.
8. Routines to create, run and destroy processes.

The first three sets require no explanation; the others will be discussed individually in subsequent sections.

It should be clear that ECL is an unusually eclectic system. This is unavoidable; a complete programming environment necessarily includes many components,

each fairly complex. There is a certain danger in this. Such a system can easily become very large, hence prohibitively expensive to implement and maintain. No less dangerous is the possibility that a system may be unwieldy for the casual users. Finally, there is the danger that the system may impose too much or the wrong kind of structure on the programmer. With each decision made incorrectly, a language system inconveniences some class of users. With many decisions to make, a system is certain to inconvenience all programmers some of the time.

In ECL, these very real dangers of an eclectic system have been avoided by judicious application of four concepts: (1) extension mechanisms, (2) sustained variability, (3) bootstrapping, and (4) system uniformity.

The first of these has been mentioned earlier. The idea is to construct a small initial system consisting mostly of powerful definition facilities for self-extension. Only the initial system—the nucleus—need be implemented and maintained by the system's creators. The rest is built on this by the programmer or programming group to suit its needs and taste. The ECL provides definition mechanisms for extension along three axes: syntax, data types, and control structures.

A second key concept, distinct from language extension, is systematic variability. That is, the deliberate provision for access by the programmer to key points at which he can control system behavior. All well-designed systems have key points of control; usually, however, these points are deeply embedded in the system either on grounds of supposed efficiency or because actions to be taken were believed to be incapable of sustaining intelligent variation. Seldom is the burial justified. Allowing programmer control over such issues provides a surprising amount of power. In ECL, three points have been singled out for attention: error and interrupt handling, input/output stream direction, and data type conversion on binding formal parameters of routines to their arguments.

Bootstrapping, i.e., using the system to define parts of itself, provides system variability at another level. In ECL, bootstrapping has been a fundamental implementation technique. The data type extension facility was used to create the system data types needed by the interpreter itself. Further, large parts of the system are coded in the language, most notably the compiler. Such system modules can be run either interpreted or compiled: the compiler, of course, is compiled by itself using the interpreter. For the system implementor, this technique avoids a large amount of machine language coding with the attendant benefits of rapid production, better system organization, and ease of change. For sophisticated system users, this bootstrapping provides

an additional point of variability: those portions of the system coded in the language are accessible to change.

The fourth concept in the ECL system is uniformity. Insofar as possible, the entire environment of the programmer is treated as a single homogeneous space without special times, cases, or preferred objects. Correspondingly, the implementor has to deal with a system notable for its lack of special cases and "funny" situations.

All data types (called *modes* in ECL) are treated equally. Each class of objects in the system has a mode; for each mode there are values of that mode; declarations are used to create variables which can be assigned values of that mode. A procedure is like any other object in this respect. It is a value, it has a mode, and may be assigned to be the value of a procedure-valued variable. Programs can be treated as data and data as programs. Programs which generate other programs are straightforward. Files (somewhat generalized) are another mode in the system, so that programs can compute the source of or sink for input/output and can arrange for arbitrary transformation of the data during transmission. Finally, there is no preferred status for the data type *mode*. A mode (e.g., *integer*) is just as legitimate a value as, say, 3.1. Hence, mode values may be computed, assigned to variables of data type *mode*, passed as arguments to routines, etc. There are a number of system-defined routines which take modes as arguments and produce new modes. Additional routines for computing modes may be defined by the programmer from these. Hence, a programming project might include all of the following (c.f. Figure 2):

1. Defining a set of routines which compute modes.
2. Writing a program which uses variables whose modes are of the class generated by 1.
3. Running the program defined in step 2 interpretively, halting, modifying and debugging it.
4. Running the routines of step 1 on input data to compute a set of modes.
5. Compiling the program of step 2 to get object code tailored to the data types computed in step 4.
6. Running the object program of step 4 on a data set.

Conceivably, this could be done in a single console session. Alternatively, these steps might be carried out over the course of several months as a large programming effort goes through the process of defining its data formats, coding and checking out its routines, metering the input profile, compiling and tuning code, and finally running. The key point is that all these

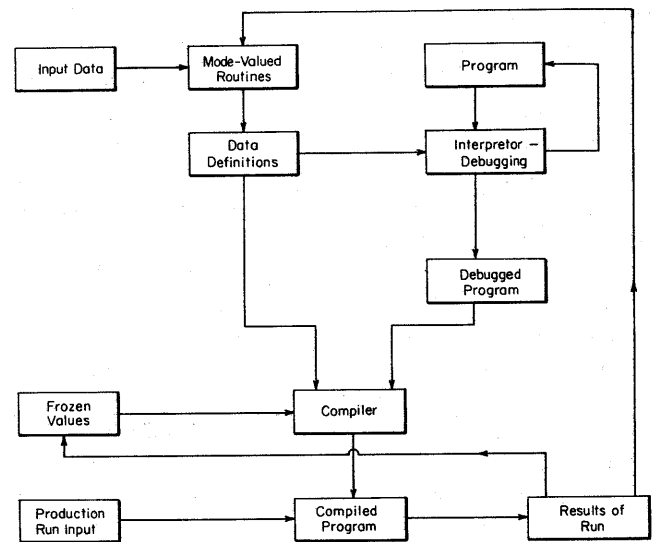


Figure 2—Program development in ECL

steps can be carried out in a single system using a common language to describe their actions.

SYSTEM FACILITIES

In this section we discuss the key facilities seen by the programmer using ECL. In the interest of brevity, we concentrate on innovative features and treat lightly those which are straightforward. In discussing the language component, we will ignore all but its extension mechanisms; in particular, we do not give its syntax or programming examples in this paper. Suffice it to say that the language is ALGOL-like in syntax, ALGOL/LISP-like in semantics and that a formal description of both syntax and semantics exists.³ Builtin data types of the language include characters, integers, reals, and Booleans; builtin operations include the usual operations on these types. A system-provided extension package adds to this the data types symbol, list and arrays of reals, integers, and Booleans along with appropriate operations.

Syntax extension

A number of proposals for syntax extension have appeared during the past few years, proposals ranging from simple macro extension schemes requiring prefix macro name triggers, to recognition of arbitrary context-free languages with complex parse-tree manipulation facilities. The technique used in ECL has two key properties: (1) it is very efficient in both parse speed

and storage required, (2) it includes specific provision for simple common additions as well as complex comprehensive changes.

The parser is a deterministic pushdown store analyzer. It scans the input stream from left to right, recording the progress of the parse in state information. At each step, the parser either reads the next lexeme and adds it to the pushdown store or it reduces the top elements of the pushdown store. In either case, it goes into a new state. In the case of a reduction, employed whenever a complete syntactic phrase has been found, a semantic action associated with the phrase class is executed. The choice of read or reduce, the reduction to be made, and the next state to be entered are recorded in a syntax table as a function of the current state, next lexeme, and top elements of the pushdown store. This table is computed by a parse table generator using a technique developed by F. DeRemer,⁴ from a syntax specification in BNF. Semantic actions augmented to each syntax rule specify the desired mapping from the parse tree into the intermediary list structure representation—IL. Each syntactic form of the source text is therefore represented by some IL list.

The interpreter and compiler treat certain IL lists (e.g., those representing a $\langle \text{block} \rangle$) specially; all others are taken as procedure or operator calls where the head of the list is the function name and the rest of the list is the set of arguments. Therefore, most augments simply map the syntactic construction into prefix form. The final element of the language specification is the definition of the function names used as prefix operators in IL.

The language may be extended by (1) adding to the *syntax specification* new syntax rules with augments, (2) defining the function names used as prefix operators in the new IL forms, thereby defining the *semantic specification*, (3) calling the *parse table generator* on the new syntax specification, and (4) switching the parser to be driven by the resulting new parse tables. In subsequent input any command, in particular any program, containing the new constructs will be analyzed employing the new syntax rules, mapped by the augments into prefix form, and executed by the associated function in the *semantic specification*. Compiling the program and the semantic specification functions will yield acceptable although not specially optimized code for the new construct.

The most common additions to the language will surely be new operators. For example, much of APL⁵ can be obtained simply by defining the appropriate array operators. While new operators could be added by using the above technique, this is needlessly complex for such a simple addition. Hence, ECL provides a special facility to handle this, making the definition

of a new operator no more difficult than the definition of a new procedure. An identifier in the language can be written *either* like a PL/I identifier (e.g., X, TEMP, FOO, COEFFICIENT) or as a sequence of special symbols (e.g., +, -, **, +←, = # >). Any identifier can be declared to be a prefix operator, an infix operator, or both. (E.g., the minus sign denotes negation as a prefix operator and subtraction as an infix operator.) An infix operator can be given an integer index from 1 to 7 specifying its binding strength.

The mechanism used to implement this facility is a simple extension of the basic analyzer; hence, operator and other extensions mesh together smoothly. The initial syntax specification includes the syntactic categories $\langle \text{prefix operator} \rangle$ and $\langle \text{infix operator}_i \rangle$ for $i=1, \dots, 7$. All operators are recognized as $\langle \text{identifiers} \rangle$ by the lexical analyzer and are handed to the parser with syntactic category $\langle \text{identifier} \rangle$. The parser changes the syntactic category to $\langle \text{prefix operator} \rangle$ or $\langle \text{infix operator}_i \rangle$ under "appropriate conditions" (e.g., for the second identifier in X**I). The parser recognizes the possibility of such an appropriate condition by means of a second set of parse tables (actually part of the symbol table) which specifies which identifiers may be used as operators and in what roles (i.e., prefix, infix_i, or prefix and infix_i). The tricky point here is distinguishing between different uses of an identifier symbol; e.g., if #@ has been declared to be both a prefix and infix operator then it may appear in:

```
# @ B      as a prefix operator acting on B,
A # @ B     as an infix operator acting on A and B,
# @ ← ...  as an identifier being assigned a new
            (operator) value.
```

The parser distinguishes between these three uses in the same way as the human reader—by local context. The read routine of the parser examines each $\langle \text{identifier} \rangle$ that can be used as an operator, checks its local context and decides how it is being used in the context, and possibly changes its syntactic type to $\langle \text{prefix-operator} \rangle$ or $\langle \text{infix-operator}_i \rangle$. The rest of the parser, in particular the part that performs reductions, is oblivious to this local transformation; it sees either an $\langle \text{identifier} \rangle$, a $\langle \text{prefix-operator} \rangle$, or an $\langle \text{infix-operator}_i \rangle$ and regards these as disjoint terminal categories.

Storage management

There are two classes of storage provided by the ECL system: (1) storage automatically allocated and freed at block entry and exit (on the *stack*) and (2) storage dynamically allocated by the program (in the

heap, using Algol 68⁶ terminology). The former is handled by well-known stack implementation techniques and requires little discussion. In providing dynamic storage allocation, however, there is a critical design decision—whether to provide automatic storage reclamation or whether to require explicit return of unused storage, e.g., by a *free* command.

A common characteristic of allocated storage is that the programmer does not, in general, know when it is becoming unused. Typically, a block is pointed to from many places, most of which are in other allocated blocks. Deciding when the last reachable pointer ceases to reference a block is therefore no simple matter. Keeping track of this at all times places a burden upon the programmer, one that may significantly complicate a program. Hence, ECL provides automatic reclamation.* Garbage collection was chosen as the implementation technique since this requires the least housekeeping storage and is guaranteed to find all unused storage. The programmer sees only a system-provided allocation function—ALLOC. Specifically, ALLOC(M) allocates an object of mode M and returns a pointer to this object. When available storage is exhausted, the allocator invokes a garbage collection.

The garbage collector is basically straightforward. A few subtle points are, however, worth mentioning. The trace phase traces all storage blocks referenced and marks all *machine words* in use using a bit map. By marking machine words, not objects, it is possible to mark only part of a block in a compound object. Garbage collection leaves untouched these parts actually referenced and reclaims the rest. The difficult point in the trace phase is the possibility, indeed almost certainty, of tracing through objects having programmer-defined mode. Given an object, the trace routine must be able to determine how big it is (so as to mark all of its words), whether or not it has pointers within it and, if so, where they are (so they can be traced). This information is calculated by internal system routines whenever a new mode is defined and is entered into tables associated with the mode. Once marking is complete, the garbage collector sweeps linearly through storage, collects all unmarked words into maximal contiguous blocks, and sorts these blocks by size into a set of linked lists forming the free storage pool. Keeping different lists for various sized blocks (currently, one list for each power of 2) speeds up subsequent allocations.

Clearly, it is best to avoid garbage collection entirely if possible. We therefore stress that ECL also provides

automatic, block-structured storage. This behaves like a normal ALGOL 60 stack, holding variables declared to reside on the stack as well as arguments to routines, and temporary results. Hence, all computation concerned with ALGOL-like objects (e.g., scalars and arrays of fixed-point and floating-point numbers) can be carried out on the stack and requires no use of the free storage mechanism.

Data type extensions

Perhaps the chief requirement of a programming language intended to serve a wide range of application areas is an equally wide range of data types or *modes*. Clearly, a language must include integers and reals for numerical computation, Booleans as the result of relational operations, and characters for headings and labels. List processing implies data objects which reference other objects, i.e., pointers. However, compiled code can be made considerably more efficient if a pointer variable may be declared as restricted in what types of objects it can point to; this introduces integer pointers, real pointers, character pointers, Boolean pointers, etc. Packed objects such as bit vectors are sometimes essential in saving core storage. A list of interesting data types could go on indefinitely.

In the face of so many diverse claimants for inclusion in a language, the only sensible solution is an extension facility: here, a mechanism for defining new modes. The language provides a few basic modes and five primitive routines for defining new modes in terms of these. The primitive mode constructors are ARRAY, PTR, STRUCT, PROC, and ONEOF; these create arrays, pointers, heterogeneous structures, procedures, and mode unions, respectively. These mode constructors are callable routines. They evaluate their arguments, perform some computation, and deliver a result having data type *mode*. The resulting modes are just as legitimate as the builtin modes. Objects of these types may be assigned values, passed as arguments to routines, returned as the value of routines, etc.

The key point of this facility is that the mode constructors compile modes in the same sense that a traditional compiler compiles procedures. That is, they calculate once, at the time a mode is created, all information about the mode that the system will subsequently need. One such computation is the storage layout for compound objects—how to represent objects of the constructed mode in the fewest possible machine words. The current algorithm produces optimal packing on almost all cases; e.g., a structure consisting of one 18-bit pointer, four 7-bit characters, three 5-bit fields, one 3-bit field and four 1-bit fields will be packed into two

* Dynamic storage management in ECL therefore differs from that of PL/I.⁷ The latter provides dynamic storage allocation but no automatic reclamation.

36-bit words.* The result of the calculation is a structure table giving the location and mode of each component in a compound object, to be used by subsequent phases of mode compilation and by the runtime routines. Another computation is preparing the tables for the garbage collector, in particular, deciding whether an object of this mode contains a pointer to be traced. The most important computation, however, is the generation of three blocks of machine code: (1) to construct objects of this mode, (2) to perform assignments to objects of this mode and (3) (for compound objects only) to select the individual components of objects of this mode. To effect construction, assignment, and selection, the interpreter executes these code bodies so that these operations are partly compiled, even from interpreted code. The compiler may either use these bodies or compile corresponding code in-line depending on whether it is optimizing space or time.

The programmer can use these mode compilation routines to define the types he needs. For example, bit vectors are defined as ARRAYs of Booleans, multi-dimensional arrays of any sort are defined by composing the function ARRAY, data processing records are STRUCTs of characters and integers, and a list of reals is constructed from blocks of identical STRUCTs each containing an integer and a pointer to the next block. Further, the programmer can define new mode-valued procedures (i.e., mode generators) in terms of the primitive routines. We anticipate a library of modes and mode-valued procedures analogous to a library of numerical algorithms.

One additional facet of the mode extension facility requires discussion. When a mode is defined using the system primitives, certain behaviors are automatically assumed. For example, if BYTE names the mode ARRAY of 8 Booleans (represented as an 8-bit object), it will be assumed that an object X of mode BYTE has 8 components which may be accessed as Boolean values by $X[I]$ for $I = 1, \dots, 8$, that assignment of one BYTE to another copies all 8 bits, and that if X is to be passed as an argument to a routine then that routine must have a corresponding formal parameter of mode BYTE. If the programmer wishes, he can override these assumptions and specify the behavior he wants. He can, for example, declare that an object X of mode BYTE is to have the following behavior:

1. X can be assigned an integer value (e.g., $X \leftarrow 73$).
If the value can be represented in 8 bits 2's

* This is, of course, entirely machine-dependent. However, the programmer never sees this packing. He deals only with objects of the language which have the right properties—e.g., access to the second 1-bit field gets the desired value. This differs from the approach taken in LISP 2⁸ where the programmer deals explicitly with the bit packing himself.

complement notation, an 8-bit assignment is made; otherwise, an error procedure P is to be called with the integer value as an argument.

2. X can be used as an argument to a routine taking an integer formal parameter, in which case sign extension is used to get a full-word value to be treated as signed integer.
3. X is to be treated as *if it* had an additional 9th component recording the number of leading 0's in its bit configuration. $X[9]$ is always interpreted as an integer count of the number of leading 0's in $X[1] \dots X[8]$ at that point in the computation.

Using this facility, the programmer can specify exactly the properties of his data objects. Encoded representation for values, variables which monitor their values, objects with "protected" fields, and the ability to represent sparse compound objects fall out as simple applications.

Compilation

A compiler can be viewed in two distinct ways. It can be taken as a device for translating programs from source representation to one which can be executed directly by some computing machine. Alternatively, it can be seen as a means for factoring a computation into two parts: that which is invariant with respect to input data and can be performed once at compile time, and that which depends on the data and is therefore postponed until run time. The second view subsumes the first and is surely the more fundamental. Translation is only one of many computations that can be factored out. Others include: evaluation of expressions at compile time, data type checking, and generic selection. The interesting problems in compilation can be best addressed by pushing the notion of factoring to take advantage of additional invariants. It is this line of approach that characterizes the ECL compiler.

A program consists essentially of a large number of variables, a few constants, and some punctuation to paste this all together. ECL carries the notion of variable somewhat farther than most languages. For example, a program may declare X to be an object of mode TRIPLE when TRIPLE is a mode-valued variable or may apply FOO to a set of arguments where FOO is a procedure-valued variable. This allows the programmer great flexibility, but presents the compiler with the problem of dealing with an unknown value of the variable. There are three possible routes it might take:

1. Attempt to deduce the value by examining the structure of the program, e.g., look for an initiali-

zation of or assignment to TRIPLE and verify that the value will not change.

2. Obtain explicit assistance from the programmer.
3. Wait until run time when the value will surely be known.

From a theoretical point of view, the first route has certain appeal. However, the inevitable undecidability results are assurance that in general one can deduce nothing; discovering subcases in which interesting deductions can be made is a significant research problem. Further, making such deductions is often a pointless task: the programmer usually knows far more about a program than could ever be deduced from examining it; he alone knows its intended function and the environment in which it is to run.

Hence, the second route is the mainstay of the compiler. In compiling a procedure P , the compiler is called with two arguments: P and a list L of all variables in P whose value is to be "frozen." P is then compiled with each variable on L replaced by the value of that variable at the point where the compiler is called. (It will be recalled that this point might be while executing another procedure or P itself.) For example, if X is declared in P to be a TRIPLE and TRIPLE is on the *frozen list* L , then the value of TRIPLE must be a mode and this mode is taken as the data type of X . Similarly, if FOO appears on L , then an appearance of $\text{FOO}(\text{arg}_1, \dots, \text{arg}_n)$ can generate code specific to the value of FOO, e.g., by in-line expansion. To treat a related case, it may be that FOO does not appear on L , but FOO is declared in P to have mode FOOMODE and FOOMODE is a variable on L . The compiler then does not have access to the value of FOO, but it does know its data type, i.e., the modes of its arguments and the mode of its result. Hence, the compiler can perform type-checking of arguments in calls on FOO and type-check the usage of its result in a larger context (e.g., $A + \text{FOO}(\text{arg}_1, \dots, \text{arg}_n)$).

Any set of variables may appear on the freeze-list L . If an operator and all its arguments are frozen (e.g., by appearance on L), then the entire function application is frozen.* By recursive application of this rule, it is possible for arbitrary complex expressions to be frozen. These can and will be evaluated during compilation. For example, if X , Y , FOO, and FUM are all on L , then

$$\text{FUM}(Y, \text{FOO}(Y), \text{FOO}(\text{FOO}(X)))$$

will be evaluated, the result replacing that expression in the code generated.

For those variables not in L , the third route remains

open: wait until run time to obtain its value. This includes "ordinary" variables as well as mode identifiers and procedure names. For example, if TRIPLE is not on L , then in a procedure with formal parameter declared to be a TRIPLE the data type is left open until the procedure is called. The compiler is governed by a consistent rule: it will compile the best code it can with the amount of information (i.e., set of invariants) given to it. This code can be anything from a single call on the interpreter (in those cases where nothing useful is frozen) to the *value* of the program (in those cases where everything is frozen). The interesting cases fall somewhere in between.

It is possible to compile a procedure dynamically during the course of some computation as values are calculated and frozen. Hence, a computation may involve reading part of the input data, compiling a program specific to that data, and running the compiled routine on the remainder of the data. Programs which periodically recompile themselves based on statistics gathered during the course of a run are an obvious application.

Errors and interrupt handling

It should go without saying that a modern programming language needs a facility for handling errors and interrupts. That is, a means for accepting asynchronous external interrupts and dealing with internal error conditions. ECL takes care of both by means of the procedure call mechanism. Every error or interrupt may be treated as if the program had explicitly called an error handling routine of its choice from the point where the error or interrupt occurred. Associated with each* error or interrupt is a procedure name (e.g., ENDOFILE, FLOATOVF, FIXOVF, etc.). When an error occurs or an interrupt comes in, the normal computation sequence is suspended at that point and a system routine ERR is entered. ERR finds the symbolic name associated with that error/interrupt condition and then checks whether there is a variable of type *procedure* valid at that point in the suspended computation. If no such variable exists, ERR types out an error message and goes into a *break routine* which preserves the state of the computation and accepts further commands from the console. If, however, there is an appropriate variable, then the associated procedure is called; so far as ECL is concerned, that is the end of the matter—any further action is the responsibility of the called routine.

* These include: end of file, fixed point overflow, floating point overflow, taking the value of a null pointer, the completion of certain I/O transactions, subscript index out of range, and timer interrupt.

* Assuming that the operator definition contains no free variables.

In the case of an external interrupt, it may be possible to handle the interrupt without regard to the suspended environment. Such an interrupt processor may perform some computation on the interrupt message, change global flags, variables, and queues, then continue with the suspended computation. However, to handle most errors and internal interrupts, it is necessary to access the environment in which the condition occurred. For example, it will frequently be useful to examine the call structure (the sequence of function calls that lead to this point) and to examine and change the values of variables in the suspended environment. ECL allows access to this information, not as a special feature offered to the error handling routine, but rather as a system facility available at all times. A stack of return points is used by ECL so as to allow recursive procedures; it is a simple matter to also stack the symbolic name of the called routine. Hence, any procedure, whether called to process an interrupt or otherwise, can obtain the symbolic name of the *I*th dynamically preceding routine (CALLER(*I*)) and can access the value of any variable in that environment (DYB(⟨variable name⟩, *I*)).

An error or interrupt routine can exit in a number of ways, depending on the cause of the interruption. GOTO *L* transfers control to the nearest enclosing label *L*; this, however, may be arbitrarily far back in the chain of calls. Since the argument to GOTO is evaluated, it is possible to use DYB to get to an arbitrary level, even one "masked" by another label of the same name; e.g., GOTO DYB(*L*, *I*) transfers control to the label *L* defined in the *I*th enclosing environment. Two other routines allow returning a computed value. For errors, CONTWITH(⟨expression⟩) continues computation with the value of ⟨expression⟩ used in place of the expression which caused the error. RETURN(⟨expression⟩, *I*) acts as if the *I*th routine back on the call chain had suddenly returned to its caller with the value of ⟨expression⟩.

In summary, this scheme provides a powerful, inexpensive mechanism giving the programmer fine control over errors and interrupts. The program is armed for a specific error or interrupt in any scope where a procedure-valued variable of the appropriate name is defined. Errors or interrupts for which the program is so armed are handled by the specific routine. Control and environmental inquiry facilities of the system provide the linguistic power needed by the routine to handle such conditions intelligently.

Control structures: paths and multiprogramming

The error/interrupt facility allows the mainline of computation to be suspended so that a subsidiary

computation can be performed to process the cause of interruption. However, this is strictly a priority situation: the interrupt routine must complete and exit before the main computation can continue. It is frequently useful to deal with subsidiary computations going on whenever there is any work to be performed, in parallel with the main computation.

ECL provides such parallel computation. In general, a job consists of some dynamically varying number of independent processes (called *paths* in ECL). What has been described thus far is the behavior of one such path. Indeed, when ECL is started, there is but one path. However, that path may create new paths and start computations on these paths, computations which in general proceed asynchronously with respect to computation on the starting path. Each path is an independent computational entity consisting of an *environment* (the call structure and variables created during this call sequence) and an *activation record* which, among other things, records the state of the path. States include suspended, waiting for some resource (e.g., I/O), and runnable. All runnable paths are parallel processes. The state of a path may be changed by a number of commands; these include SUSPEND some path, WAIT some period of time, and the Dijkstra P and V semaphores⁹ for synchronization among paths. All paths have a certain portion of their environment in common—potentially, any allocated storage. Hence, it is possible for two or more paths to reference common data, e.g., a buffer, a set of flags, or a message queue. This, coupled with the P and V semaphores, allows the conventional sort of cooperating sequential processes to be established.

The really interesting aspects of the ECL path facility lie, however, in its ability to host nonconventional multiprogramming, in particular, control regimes not explicitly anticipated by its designers. That is, like many other facilities in ECL, the multiprogramming mechanism is extensible. As with other extension facilities, that for multiprogramming consists of a set of primitives and a framework for combining them. Primitive operations include creating a path, setting up a function to be executed in a created path, running a path, deleting a path, accessing and changing the value of a variable in some other path, and making a copy of a path. The basic framework is provided by a distinguished path—the *control interpreter*. This is unique in two respects: (1) timer interrupts pass directly to it; (2) there is a control primitive—CIA—by which other paths can call for the execution of an arbitrary procedure in the environment of the control interpreter and wait for the result.

There is a program which runs in the control interpreter path and acts as the central control of ECL.

Basically, its functions are to handle I/O requests, arrange for running the other paths, and handle coordination between paths. This program is written in the language using the primitives mentioned above. For example, to perform path scheduling, a queue of runnable paths is maintained; when the timer interrupt comes into the control interpreter, the path that was running is put at the end of the queue, a new path is chosen from the runnable queue by the *scheduler*, and the *start-path* primitive is executed to run the new path. The *scheduler* is also a routine written in the language. Currently, it simply chooses paths in FIFO order. However, the programmer may redefine the scheduler by substituting his own routine for the system-provided one. Hence, such refinements as a priority system, either simple or with dynamically changing priorities, can be readily added.

Other control activities are equally easy to program. For example, a Dijkstra semaphore is a language-defined data structure consisting of an integer count and a queue of paths (also a defined data type) waiting on this semaphore. The P and V operations are implemented by using CIA primitive to transfer into the environment of the control interpreter where the necessary queues can be safely modified.

With the framework provided, it is straightforward to implement most of the known control structures, e.g., coroutines, multiple parallel returns, cooperating sequential processes and fork/join structures. Further, since ECL leaves its control structures open to change, it will be possible to develop, as needed, a variety of other control regimes.

SUMMARY

The ECL programming system has been designed to provide an environment conducive to effective programming. To this end, it contains a language with comprehensive data types, operators, control structures, and storage management facilities. It allows interactive program composition and debugging with smooth

transition to efficient compiled code. Most important, it allows the programmer to tailor this environment to suit his needs.

ACKNOWLEDGMENTS

It is a pleasure to acknowledge the help of B. Brosgol, B. Byer, T. Cheatham, B. Holloway, and C. Prenner.

REFERENCES

- 1 B WEGBREIT
The treatment of data types in ECL
Technical Report 4-71
Center for Research in Computing Technology
Harvard University Cambridge Massachusetts May 1971
- 2 B WEGBREIT
Compactifying garbage collection in the heap
Technical Report 5-71
Center for Research in Computing Technology
Harvard University Cambridge Massachusetts June 1971
- 3 B WEGBREIT
Studies in extensible programming languages
ESD-TR-70-297
Harvard University Cambridge Massachusetts May 1970
- 4 F L DE REMER
Practical translators for LR(k) languages
Ph.D. thesis
Electrical Engineering Department MIT Cambridge
Massachusetts October 1969
- 5 IBM
APL/360 user's manual
GH 20-0683-1
- 6 A VAN WIJNGAARDEN *et al*
Report on the algorithmic language ALGOL 68
Mathematisch Centrum Amsterdam MR 101 February
1969
- 7 G RADIN H P ROGWAY
Highlights of a new programming language
Communications of the ACM Vol 3 January 1965
- 8 P S ABRAHAMS *et al*
The LISP 2 programming language and system
FJCC Vol 29 1966
- 9 E W DIJKSTRA
Co-operating sequential processes
In *Programming Languages* edited by Genuys Academic
Press New York 1968

The “single-assignment” approach to parallel processing*

by DONALD D. CHAMBERLIN

IBM Thomas J. Watson Research Center
Yorktown Heights, New York

INTRODUCTION

Parallel processing systems—computer systems in which more than one processor is active simultaneously—offer potential advantages over uniprocessor systems in terms of speed, flexibility, reliability, and economies of scale. However, they pose the problem of how multiple processors can be organized to cooperate on a given problem without interference. Various solutions have been proposed to this problem.^{1,2,3,4} Some systems require a programmer to assign units of work to the various processors, while other systems perform this assignment automatically; in some systems, the processors are linked closely together, while in others the processors are nearly independent. The system to be described here automatically detects opportunities for parallel processing in programs written in a specific, high-level language. Parallelism is detected on a very low level—even within a single algebraic expression. The system consists of many independent, asynchronous processors, all active at once in processing a single program.

In a paper presented at the 1968 Spring Joint Computer Conference⁵ Larry Tesler and Horace Enea proposed the concept of “single-assignment” programming languages. In a single-assignment program, statements do not necessarily execute in the order in which they appear; rather, each statement executes as soon as all the variables it needs are defined. In order that each statement be triggered at a well-defined time, it is required that each variable be assigned a value only once during the execution of a program. In such a program, there is no “flow of control” in the conventional sense; rather, the sequencing of statements is determined by the data flow, as some state-

ments assign values to variables which are needed by other statements. If many statements simultaneously have all their needed variables defined, all the statements may be executed in parallel.

This paper describes a single-assignment language called SAMPLE (for Single-Assignment Mathematical Programming Language), and a parallel processing system to implement the language. SAMPLE was originally inspired by Tesler’s language COMPEL; however, it includes quite different facilities for iteration, input/output, and other features not found in COMPEL. All considerations of implementation are original to this paper.

THE LANGUAGE

SAMPLE resembles as closely as possible a conventional high-level language such as ALGOL. It employs the left arrow (\leftarrow) as an assignment operator, and allows use of the conventional arithmetic and logical operators and parentheses in the construction of expressions. However, all SAMPLE programs must obey the single-assignment constraint: No variable may be assigned a value more than once during the execution of a program. SAMPLE recognizes two data types: Real numbers and tuples, which are ordered sets of numbers or of other tuples. By nesting tuples inside each other, the programmer can implement arrays, trees, or “structures” as in PL/I. Tuples are denoted by lists of their elements, enclosed between \langle and \rangle . For example, the array

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

might be represented by the nested tuple

$$A = \langle \langle 1, 2, 3 \rangle, \langle 4, 5, 6 \rangle, \langle 7, 8, 9 \rangle \rangle.$$

* This work was done while the author was with Digital Systems Laboratory, Electrical Engineering Dept., Stanford University, and was supported by a National Science Foundation Graduate Fellowship.

Elements are accessed by means of the subscripting arrow \downarrow , which is grouped from the left if it occurs multiple times. The first element of a tuple has subscript one unless otherwise specified. In the above example $A \downarrow 2 \downarrow 3$ is the number six. The reserved word TO generates a tuple containing successive integers; for example, $\langle 11 \text{ TO } 14 \rangle$ is the same as $\langle 11, 12, 13, 14 \rangle$.

The reserved words FIRST and LAST yield the subscript number of the first and last element of a tuple, respectively. For example, LAST $\langle 11, 12, 13, 14 \rangle$ has the value four.

Arithmetic and logical operators may be used between two tuples or between a number and a tuple, in which case they operate element-by-element.

Examples:

$$\langle 1, 2, 3 \rangle + \langle 2, 3, 4 \rangle = \langle 3, 5, 7 \rangle$$

$$\langle 1, 2, 3 \rangle + 2 = \langle 3, 4, 5 \rangle.$$

Assignments may be made to individual elements within a tuple, provided that the single-assignment constraint is observed. When a tuple variable is to have its elements assigned one at a time, an additional "bounding" statement must be included to inform the system that the variable is a tuple, and giving its first and last subscript numbers (which may be expressions). In the following example, A is defined to be a tuple having subscripts ranging from one to three, and its elements are assigned values:

A IS TUPLE (1, 3);

A \downarrow 1 \leftarrow 5;

A \downarrow 2 \leftarrow 13.5;

A \downarrow 3 \leftarrow (X+Y)/Z;

Rather than writing a separate bounding statement, the programmer may choose to state the bounds of a tuple's subscripts in the same statements which assign values to its elements, as in

A \downarrow 1 OF (1, 3) \leftarrow 5;

If the lower subscript bound is omitted, it is taken to be one. The "OF" clause may be used more than once in a statement, once for each subscript. Thus,

A \downarrow I OF L \downarrow J OF L \leftarrow 50;

means:

1. A is a tuple whose subscripts range from one to L
2. A \downarrow I is a tuple whose subscripts range from one to L
3. A \downarrow I \downarrow J is assigned the value 50.

SAMPLE has no conditional or unconditional

branches, because it has no flow of control. However, it has a conditional expression. The following statement assigns to SWITCH the value A if $X=Y$, otherwise the value B:

SWITCH \leftarrow IF $X=Y$ THEN A ELSE B;

SAMPLE has block structure, which enables the programmer to declare and use a name in an inner block without danger of duplicating a name used elsewhere. However, no storage allocation occurs on block entry; in fact, "block entry" is undefined since statements execute in an unpredictable order.

The unpredictable order of statement execution also requires some special provisions for input and output, since the programmer cannot know the order in which quantities will be read or written. Conceptually, an I/O medium is provided in which all inputs are simultaneously available, each associated with a "tag" (each tag is a unique integer). The statement READ (A, 1) means "read into variable A the input quantity associated with tag one." WRITE (B, 2) means "write the value of B into the I/O medium and associate it with the tag two." Post-processing can be done on the I/O medium to produce the desired output document.

SAMPLE allows the programmer to define functions, which are pieces of code which may be called by name from various places in the program. Functions may have parameters and may be recursive; however, a function must obey the single-assignment constraint internally, and must return exactly one value and have no side effects.

The most difficult facility to provide in a single-assignment language is iteration, which, by its nature, tends to assign values to the same variables repeatedly. Since SAMPLe is intended for parallel processing, we distinguish two types of iteration: (1) a set of actions which may be taken simultaneously, and (2) a set of necessarily sequential actions. For simultaneous iteration, we provide a convention which obeys the single-assignment constraint. A statement containing a tuple name enclosed in single quote marks behaves exactly as though it were many statements, one with each element of the tuple substituted for the quoted name. If more than one different quoted name appears, a copy of the statement is generated for each possible way of substituting a tuple element for each quoted name. The order of execution of the copies is determined by the readiness of their respective input values. For example, if the programmer writes

I \leftarrow $\langle 1, 2 \rangle$;

J \leftarrow $\langle 1, 2 \rangle$;

A \downarrow 'I' \downarrow 'J' \leftarrow B \downarrow 'J' \downarrow 'I';

the last statement behaves exactly as though it were

```
A ↓ 1 ↓ 1 ← B ↓ 1 ↓ 1;
A ↓ 1 ↓ 2 ← B ↓ 2 ↓ 1;
A ↓ 2 ↓ 1 ← B ↓ 1 ↓ 2;
A ↓ 2 ↓ 2 ← B ↓ 2 ↓ 2;
```

The SAMPLE facilities for sequential iteration abridge the single-assignment property; they look almost exactly like ALGOL FOR and WHILE loops. Examples:

```
FOR I ← 1 STEP 1 UNTIL 10 DO
  (loop body)
END

WHILE X = Y DO
  (loop body)
END
```

For the purposes of the external program, the entire loop with all its iterations looks like a single statement, and any variables assigned values in the loop are not considered to be "ready" until the final iteration is complete. Within the loop body, the order of execution of statements (and nested loops) is governed by the readiness of their data, according to the single-assignment property. Each iteration of the loop is not begun until the previous iteration is complete. Within the loop, any variable name X means "the value which is being computed for X during this iteration of the loop" whereas OLD X means "the value which was computed for X in the previous iteration." Each loop may have an initialization section, which assigns the value to be used for OLD X (or any other OLD variable) during the first iteration, as follows:

```
INITIAL X ← 0;
```

Shown below is a SAMPLE program which reads a matrix A (of arbitrary size and shape) and reduces it to upper diagonal form by a process of Gaussian elimination. Rows and columns are assumed to begin with subscript one.

```
BEGIN
  READ (A, 1);
  COMMENT: A IS A TUPLE OF TUPLES
  REPRESENTING A MATRIX. EACH ELE-
  MENT TUPLE IS A ROW;
  L ← LAST A;
  COMMENT: L IS THE NUMBER OF THE
  LAST ROW;
  FOR I ← 1 UNTIL L-1 DO
    COMMENT: DO THE LOOP BODY FOR
    EACH ROW EXCEPT THE LAST;
```

```
INITIAL B ← A;
J ← (1 TO I);
K ← (I+1 TO L);
FACTOR IS TUPLE (I+1, L);
FACTOR ↓ 'K' ← OLD B ↓ 'K' ↓ I/OLD
  B ↓ I ↓ I;
COMMENT: FOR EACH ROW LOWER
  THAN THE PRESENT ONE, WE HAVE
  COMPUTED THE NECESSARY FACTOR.
  WE NOW CONSTRUCT THE NEW B
  FROM THE OLD B BY ROW OPERA-
  TIONS;
B IS TUPLE (1, L);
B ↓ 'J' ← OLD B ↓ 'J';
COMMENT: THE NEXT STATEMENT
  DENOTES ARITHMETIC OPERATIONS
  BETWEEN ROWS;
B ↓ 'K' ← OLD B ↓ 'K' - OLD B ↓ I *
  FACTOR ↓ 'K';
END
WRITE (B, 2);
COMMENT: ONLY THE FINAL VALUE OF B
  (THE FINISHED MATRIX) IS WRITTEN;
END
```

IMPLEMENTATION

Implementation of SAMPLE consists of two steps: Compilation and execution.

Compilation is done by a conventional method: The program is parsed according to a phrase structure grammar, and appropriate machine-language instructions are emitted during the parsing process. The compiler generates "temporary" variables as needed to ensure that the single-assignment property is preserved in the emitted code. For example, in compiling the expression

$$X \leftarrow (A * B) + (C * D);$$

the compiler would generate temporary variables T1 and T2 and emit the following instructions:

1. T1 ← A * B
2. T2 ← C * D
3. X ← T1 + T2

During the execution phase, instructions (1) and (2) might execute simultaneously, defining the values of T1 and T2, which in turn would release instruction (3) for execution. The compilation process, which could be implemented on a conventional machine, is described in more detail in a Stanford Ph.D. thesis.⁶ SAMPLE could not conveniently be used as the language in which

its own compiler is written, because it lacks facilities for manipulating character strings.

It has been proposed that the SAMPLE compiler might generate names, relieving the programmer of the necessity to invent a different name for every different assignment of a variable. However, this would require the compiler to make assumptions about the order in which statements are to be processed, and so would be contrary to the principle of single-assignment programming.

A hardware organization is proposed for executing compiled SAMPLE programs. The system has three passive storage units:

1. *The Instruction Store*

This unit contains the machine instructions emitted by the compiler. Each instruction has an opcode, an output operand, up to three input operands, and certain link fields as described below. Each input operand has a "ready" bit; the instruction cannot be executed until all the ready bits are on.

2. *The Data Store*

This unit contains data used during execution of the program. Each cell contains a space for the value of a variable, and a pointer to some instruction which is waiting for that variable as an input operand. If there are many such instructions, they are organized into a linked list, each instruction pointing to the next by means of special link fields in the instructions. Thus, when a given variable becomes ready, all instructions waiting for it can be notified by following the linked list. The linked list is created by the compiler (or by the action of certain instructions at run time).

A number can be stored in the "value" field of a single data cell. If the variable to be stored is a tuple, the cell contains a notation of the size of the tuple, and a pointer to where the first element is stored. The elements are stored in a set of consecutive cells, one element to a cell. Each element may itself be a tuple which points to a set of elements of its own.

3. *The Ready List*

This unit contains copies of all instructions which are known to be ready for execution, in the sense that all their input operands are defined.

Execution of the program is carried out in parallel by many independent processors. Each processor re-

peatedly executes the following Basic Instruction Routine:

1. The processor fetches from the Ready List an instruction which is ready to be executed.
2. The processor fetches from the Data Store the input operands of the instruction, and performs the indicated operation on them.
3. The processor writes the resulting output operand into the Data Store. In the same storage access cycle, it obtains the pointer to an instruction which is waiting for the newly-ready cell, if any.
4. The processor follows the linked list of instructions which are waiting for the newly-ready data cell. For each such instruction, it does the following:
 - a. It turns on the ready bit of the newly-ready operand.
 - b. If all ready bits are now on, it copies the instruction into the Ready List.
 - c. It obtains the link to the next instruction on the waiting list.

Because many processors are simultaneously making access requests to the three storage units, each is organized into many banks, and the cell addresses within the unit are interleaved among the banks. In a given storage cycle, each bank can satisfy only one access request; however, two processors making simultaneous requests of two different banks may both be satisfied.

Certain features of SAMPLE make it necessary that additional machine instructions be generated at run time. One such feature is the ability to define and call functions. The compiler produces, from the function definition, a template of machine instructions, with certain operands left "blank." Then, when the function is called at run time, a special CALL instruction makes a new physical copy of the template, filling in the blanks with the real parameters of the function call, and releases the newly copied instructions for execution. The original template is preserved and used for other calls. Because of the single-assignment constraint, each new copy of the function template must have a completely new set of memory cells allocated for its temporary variables; these new cells are allocated by the action of the CALL instruction.

Another feature requiring run-time generation of instructions is the parallel iteration feature, in which a statement containing a quoted tuple name behaves like many statements, one for each element of the tuple. In general, the size of the quoted tuple is not known at compile time, and so it is not known how many copies of the statement are to be made. Again,

the solution is to make a template of all the instructions compiled from the statement. At run time, when the quoted tuple is defined, a special EXPAND instruction is triggered, which expands the template into the required number of copies and fills in the addresses of the tuple elements in the appropriate places.

A third language feature requiring special implementation is the loop. Once again, the compiler generates a template of instructions corresponding to one copy of the loop. At run time, a physical copy of the template is made and released for execution. At the same time, a special REPEAT instruction is generated, whose function is to sense when the most recent copy of the loop is completely executed, then test the continuation condition and, if it passes, generate a new copy of the loop template (complete with its own REPEAT instruction). In order for the REPEAT instruction to sense when all the loop instructions have executed, it must have a dummy input operand which becomes ready only when the output operands of all the loop instructions are ready. This is accomplished by means of a tree of NOP instructions, whose only function is to make the readiness of the dummy REPEAT operand dependent on the readiness of all variables defined in the loop.

Loops also require introduction of the concept of "levels of readiness." A variable defined in a loop may be "ready" to instructions which are implementing the current copy of the loop, but "not ready" to instructions outside the loop, which must wait for all loop iterations to be complete before they can use the variable. Therefore, all instructions in the instruction store have a "level" field, which describes the level of loop nesting at which the instruction is found, and each cell in the data store has a field describing its level of readiness. The ready bit of an instruction is not turned on unless the corresponding operand is ready on the level of the instruction (or on an outer level of nesting).

EXPECTED PERFORMANCE

We expect that, for some class of problems, the SAMPLE type of organization has a speed advantage over a conventional uniprocessor, despite its wastage of storage accesses on overhead functions such as updating ready bits. The speed advantage arises from the ability to overlap multiple memory accesses into the same memory cycle. However, the SAMPLE system requires a costly replication of processors and memory banks, and its total storage requirements for a given unit of work are expected to exceed those of a conventional processor by a large factor. The increased storage

requirements are due to the following causes:

1. Storage is required for such "overhead" items as pointers, links, ready bits, and the Ready List.
2. Because the processors communicate only through memory, they cannot store temporary results in internal registers, but must use memory cells for this purpose.
3. The single-assignment property dictates that each storage cell is used only once in a program. Therefore, although SAMPLE processing is compacted in time, it is correspondingly "spread out" in memory space. This trading off of time for memory space may be an inevitable consequence of parallel processing.

Because of its wasteful use of storage, single-assignment processing is not considered to be a cost-effective method of computing at the present time. Its feasibility would be improved by a large reduction in the cost of random-access storage, or by development of a means of reusing storage locations during processing of a program. The problem of deciding when to free a storage cell for reuse is complicated by the fact that, in processing a SAMPLE program, new instructions are generated at run-time. Thus, although all instructions referencing a particular cell may have been executed, there is no assurance that more such instructions will not be generated at a future time, and so the cell cannot be released.

EXAMPLE

As an example of the behavior of the proposed system, programs were written to multiply together two square matrices, in SAMPLE (see Appendix A) and in IBM System/360 Assembler Language⁷ (see Appendix B). The 360 program was written in such a way as to minimize memory accesses by storing temporary results in registers. Behavior of the two programs was simulated in detail, on a memory-cycle level, for 2×2 and 3×3 matrices. The results were compared on two bases: (1) total memory cycles required to execute the program, and (2) total bits of storage required for program, data, Ready List, and all working areas. Each 360 memory cycle resulted in exactly one memory access for fetching instructions or data or storing results. In the SAMPLE system, memory accesses were used not only for these purposes but also for "overhead" functions such as updating ready bits. However, in the SAMPLE system, one memory cycle might result in many memory accesses made by different processors to different storage banks. Three parameters limited the

ability of the SAMPLE system to overlap storage accesses in this way: (1) the number of processors, (2) the number of storage banks, and (3) the number of ports through which a processor may make an access request (effectively, the number of access requests which may be made by a single processor in the same cycle). Two SAMPLE systems were investigated: (1) an unlimited system, in which there were indefinitely many processors, each with an unlimited number of ports, and each individual storage address in the instruction store or data store is considered to be its own "bank"; (2) a limited system having ten processors, each with four ports, and in which the instruction and data stores each have their addresses interleaved among ten banks. The simulation results are shown in Figures 1 and 2. The results of the example are consistent with the expected performance described above.

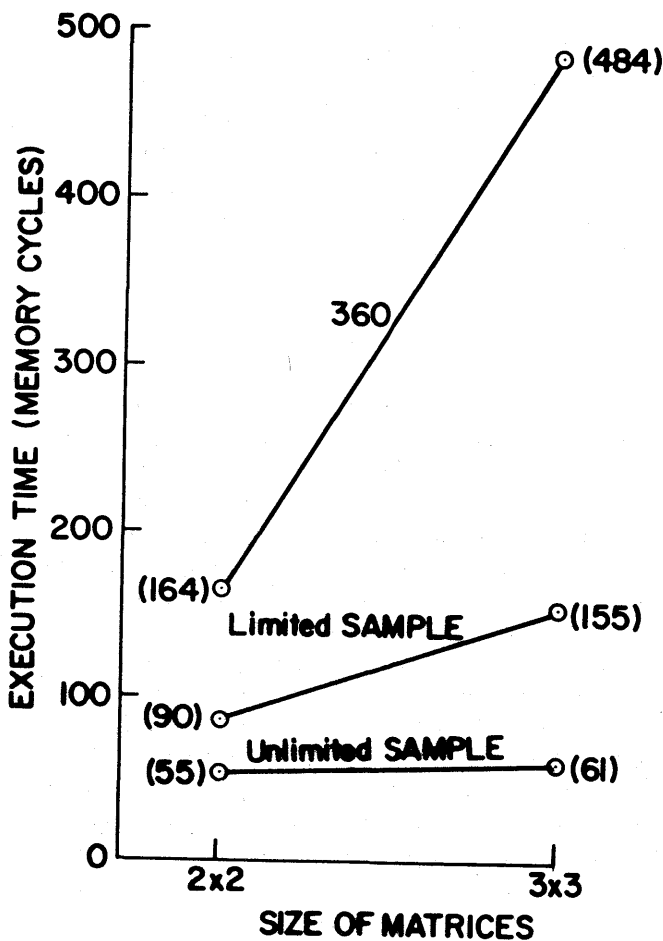


Figure 1—Execution time comparison, 360 vs. SAMPLE

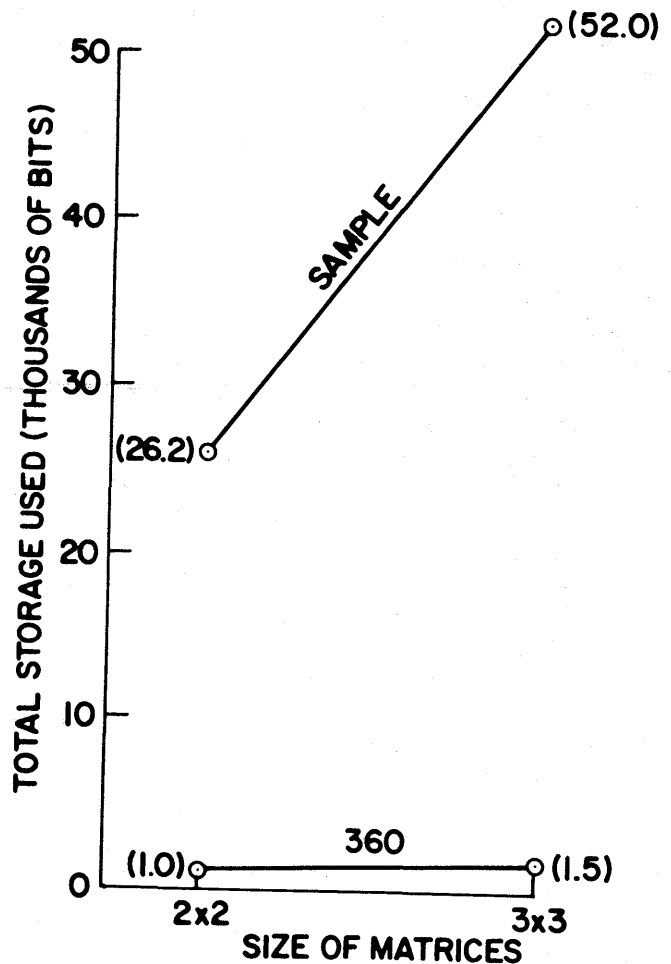


Figure 2—Storage usage comparison, 360 vs. SAMPLE

REFERENCES

- 1 J P ANDERSON
Program structure for parallel processing
Communications of the ACM Vol 8 No 12 December 1965
- 2 G H BARNES R M BROWN M KATO
D J KUCK D L SLOTNICK R A STOKES
The Illiac IV computer
IEEE Transactions on Computers Vol C-17 No 8
August 1968
- 3 H W BINGHAM D A FISHER E W REIGEL
Automatic detection of parallelism in computer programs
Burroughs Corporation Technical Report TR-67-4
November 1967
- 4 H S STONE
One-pass compilation of arithmetic expressions for a parallel processor
Communications of the ACM Vol 10 No 4 April 1967
- 5 L G TESLER H J ENEA
A language design for concurrent processes
Proceedings of the 1968 Spring Joint Computer Conference

6 D D CHAMBERLIN
Parallel implementation of a single assignment language
 PhD Thesis Electrical Engineering Department Stanford
 University Stanford California 1971
 7 *IBM System/360 principles of operation*
 IBM Publication No A22-6821 February 1966

APPENDIX A

SAMPLE MATRIX MULTIPLICATION PROGRAM

This program multiplies square matrices A and B (which may be of any size) to form the product C:**
 BEGIN

```

L ← LAST A;
I ← (1 TO L);
J ← (1 TO L);
K ← (1 TO L);
COMMENT: DO ALL MULTIPLICATIONS
IN A SINGLE STEP;
T ↓ 'I' OF L ↓ 'J' OF L ↓ 'K' OF L ←
A ↓ 'I' ↓ 'K' * B ↓ 'K' ↓ 'J';
COMMENT: NOW ADD UP THE PRODUCT
ELEMENTS;
C ↓ 'I' OF L ↓ 'J' OF L ← + T ↓ 'I' ↓ 'J';
END
    
```

APPENDIX B

IBM SYSTEM/360 MATRIX MULTIPLICATION PROGRAM

This program accepts 2x2 matrices in row-major order in areas A and B, and leaves their product in row-major order in area C. To convert to multiply N x N matrices, simply enlarge areas A, B, and C to N² words each, and change the constant N4 to contain 4 * N.

** The unary operator + in the last statement yields the sum of the elements of the tuple T ↓ I ↓ J.

```

* REG. 0 WILL CONTAIN 4 * N
* REG. 1 WILL CONTAIN 4 * I
* REG. 2 WILL CONTAIN 4 * J
* REG. 3 WILL CONTAIN 4

A    DS  4F
B    DS  4F
C    DS  4F
N4   DC  F'8'

LA   12,4      PUT 4 IN R12
L    0,N4      PUT 4*N IN R0
LR   1,12      SET I=1 (4*I=4)
ILOOP LR 2,12  SET J=1 (4*J=4)
JLOOP LR 5,1
MR   4,0
AR   5,2      R5 NOW CONTAINS
                                DISPLACEMENT
                                (I,J)
SR   10,10    ZERO R10
LR   3,12      SET K=1 (4*K=4)
KLOOP LR 7,1
MR   6,0
AR   7,3      R7 NOW CONTAINS
                                DISPLACEMENT
                                (I,K)

LR   9,3
MR   8,0
AR   9,2      R9 NOW CONTAINS
                                DISPLACEMENT
                                (K,J)

LE   11,A(7)  LOAD A(I,K) INTO R11
ME   11,B(9)  MULTIPLY
                                A(I,K) * B(K,J)
AER  10,11    ADD PRODUCT TO R10
AR   3,12      INCREMENT K
CR   3,0       IF K <= N,
BC   12,KLOOP GO TO KLOOP
ST   10,C(5)  STORE R10 INTO C(I,J)
AR   1,12      INCREMENT J
CR   2,0       IF J <= N,
BC   12,JLOOP GO TO JLOOP
AR   1,12      INCREMENT I
CR   1,0       IF I <= N,
BC   1,ILOOP  GO TO ILOOP
    
```


MEANINGEX—A computer-based semantic parse approach to the analysis of meaning*

by DAVID J. MISHELEVICH**

*The Johns Hopkins University School of Medicine
Baltimore, Maryland*

INTRODUCTION

It is the purpose of this paper to look at the semantic analysis of a subset of natural English text, namely the simple noun phrase, and present the theoretical basis for and the implementation of a semantic analyzer called MEANINGEX. A "simple" noun phrase is defined as a noun modified by adjectives and/or prepositional phrases. Throughout the paper, examples will be given from medical record text because of my own orientation and because the desirability of semantic analysis of specific types of phrases provided the motivation for my study of meaning. I look at the basic operational question to be as follows: "How can statements with the same meaning, but which are said in different words be transformed to an identical form?" Thus the basic object of the process is to make similar things fall together.

Looking at the set of medical records of a hospital or physician as a whole, the set forms a costly and hard-won body of medical knowledge from which information for both individual patient care and for research purposes can be retrieved.

As in any document which contains many, many words, organization is a requirement for the purpose of rapid retrieval. One formal scheme for producing a well organized patient record results in the "PROBLEM-ORIENTED MEDICAL RECORD" of Weed.^{1,2}

The record consists basically of a numbered list of

problems (which may or may not be diagnoses) followed by progress notes and flow sheets of laboratory values or other measurements keyed to the problems.

An important feature for the generalization of the approach to semantic analysis is that the individual problems are essentially the same in format and content as statements of diagnoses, symptoms or abnormal laboratory findings in medical and pathological records which are not in the form of the "Problem Oriented-Medical Record."

The striking syntactic feature about this highly important problem list is that, in the vast majority of instances, the statements are noun phrases. The availability of a valuable corpus of medical record text as the subset of natural language to be analyzed first, before completely free text, justified the limitation of this study in semantic analysis to noun phrases.

THEORETICAL BASIS FOR SEMANTIC ANALYSIS

Semantics has been primarily studied by philosophers. Rather than the development of a unified, applicable analysis scheme, the approach to semantics has been largely descriptive with example and counter-example.

The major thrust of recent philosophical work in semantics has been due to Dr. Jerrold Katz and his coworkers. Katz points out that there are three components to the linguistic description of a natural language: syntactic, semantic and phonological.³ They define the linguistic description of a natural language as "attempt to reveal the nature of a fluent speaker's mastery of that language" (Ibid, p. 8). Basically, then, we are concerned with the act of communication. Specifically, we are interested in the means by which one physician communicates with other physicians through the medium of the medical record. We desire to make relatively minimal restrictions on his com-

* This investigation was supported in part by National Institute of General Medical Sciences Special Fellowship No. 5-F03-GM-42, 816-02. The computing was done in the computing center of the Johns Hopkins University School of Medicine supported in part by a research grant of the Control Data Corporation. This paper is based on a dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at the Johns Hopkins University.

** Present address: National Educational Consultants, 711 St. Paul St., Baltimore, Maryland 21202.

munication and avoid such practices as the enforced use of handbooks of standard terms or numerical codes.

Katz and Fodor⁴ and Katz and Postal³ outlined a model for a semantic theory. A "projective device" was described which consisted of two components: a dictionary and set of projection rules which assign a semantic interpretation to the output produced by the syntactic process. It is required that every sense that a term can take on in any sentence be covered. Katz and Postal³ require that the entries in the dictionary have "normal form" with the following components:

1. Syntactic Markers
2. Semantic Markers
3. Distinguishers (optional)
4. Selection Restrictions (optional)

MEDICAL INFORMATION RETRIEVAL SYSTEMS

Retrieval of information in the form of diagnoses has been a very important objective in medicine. The major systems currently used are those involving numerical codes (SNDO=Standard Nomenclature of Diseases and Operations,⁵ IC=International Classification of Diseases and Operations,⁶ ICDA=International Classification of Diseases, Adapted,⁷ and SNOP=Standard Nomenclature of Pathology⁸) although there is a move (e.g., Reference 9) toward standard terms uncoded into numbers (Reference 10), such as the Current Medical Terminology.¹¹

Other medical systems not relying on numerical codes coded by the user can be broken down into two classes. The first is the synonym approach¹²⁻¹⁴ and the second is the syntax-oriented approach (the ACORN system, see References 15 and 16).

COMPUTER-ORIENTED RELATED AREAS

Four computer-oriented types of programs depend very heavily on the general field of semantic analysis. They are: Machine Translation, Question and Answer Systems (part of artificial intelligence), Content Analysis and Bibliographic Retrieval.

Machine translation

The 1950s brought forth the hope for and attempts to realize machine (or mechanical) natural language

translation from one language to another. While the results were largely disillusioning even with postediting, some questions are related to the present problem. With copious postediting, machine translation cost more than human translation.

Question and answer systems

Question answering systems must have semantic talents. Simmons has reviewed the area twice¹⁷⁻¹⁹ and indicated that there have been essentially two generations of such systems with a fuzzy dividing line. One difference has been that most first generation systems have had fixed data bases from which information can be retrieved while second generation systems are more likely to be able to accept revisions to the information structure by the individual asking the question. With rare exceptions (e.g., see Reference 20) the systems have been quite limited in the English subset involved. Bibliographic retrieval is a related area, since the search request is actually a formally stated question. Giuliano,²¹ commenting on the first Simmons article, felt that there were many reasons for being pessimistic since the semantic analysis in general was restricted to "almost trivial subject areas" such as kinship relationships,²² baseball²³ and uncomplicated geometric figures.²⁴ Some improvements were made by the time of writing of the second review article as we shall see below.

It is beyond the scope of this paper to cover such a rich field exhaustively. The area can be broken down into list-structured data base systems (BASEBALL,²³ SAD SAM²²), text-oriented systems (PROTOSYN-THEX^{20, 25}), logical inference systems (SIR²⁶⁻²⁷ and STUDENT²⁸⁻²⁹), belief system simulation³⁰⁻³⁴ and semantic memory.³⁵⁻³⁷

Content analysis

The major thrust in content analysis has been using a tool called "The General Inquirer,"³⁸⁻³⁹ a computer-oriented system for which a user's manual has been supplied.⁴⁰

Bibliographic retrieval

Bibliographic retrieval or automatic indexing is a question answering system in that the question is, "Please retrieve all references which are relevant to my search request in a given field."

The most ambitious test of retrieval techniques has been with regard to the SMART system developed by Salton and his coworkers.⁴¹⁻⁴⁶

REQUIREMENTS FOR A SEMANTIC ANALYZER

What, then, are the requirements which a semantic analyzer should be able to meet? They are in summary:

1. Application of a "semantic transformation" so that statements which are given in different words but are recognized as meaning the same thing in a given contextual communication are reduced to an identical form.
2. Effective selection of sense of meaning
3. Resolution of ambiguity
4. Deletion of contradiction
5. Ability to handle the problem of specificity

The approach to MEANINGEX will be stated, the implementation described and results on typical medical record text statements exhibited. We then shall have the opportunity to compare the output of the MEANINGEX analyzer to the above criteria and evaluate its performance.

THE SEMANTIC PARSE APPROACH TO MEANING ANALYSIS

As opposed to the selector scheme of Katz et al. with the subsequent combination of appropriately selected lexical paths, I have chosen the more efficient representation of the modification of a head term in which both the selection and combination of meaning are integrated processes. I call this a semantic parse or composite selector-modifier system. Not just the attributes of a word [e.g., (Human), (Animate), etc. of the Katz, Fodor and Postal model] are displayed. The head term is the noun in the noun phrase statement which forms the root in the tree-structured semantic analysis of that statement. Any arbitrary modification of a head term by semantic markers is permitted. The appropriate markers for the text being analyzed are automatically selected. The use of a tree structure means that the modifier of a term can be in turn

modified to any level. Thus we have schematically:

```

HEAD TERM
  MODIFIER A
    MODIFIER A1
    MODIFIER A2
      ⋮
    MODIFIER B
      ⋮
  Etc.

```

Note that typographical indentation is and will be employed as the means by which hierarchical structure is displayed. The modifiers are general properties such as anatomical or functional considerations.

I call the tree structure a semantic parse or "sparse" because the nodes are composed of semantic markers as opposed to syntactic ones. Note that Raphael²⁷ has used the term "semantic parse" to denote the phase of extracting relational information from English text in his SIR program (see above). Since the MEANINGEX modifications are at least informal relations, the present use is certainly related. Quillian³⁵⁻³⁶ feels that his dictionary view of encoding definitions using the concept of modifications to a word represents at least in part a "parse." This use is also consistent, perhaps more so. The non-hierarchical, linear presentation of the node contents in descriptor form after the tree structure has been constructed is called the "end-sparse."

An important point is that we are dealing with a functionally oriented analyzer. It has generative properties as well since the input of a single head term will generate an entire skeletal structure. The analysis is performed, however, with regard to the rest of the text given. These concepts will become clearer through the development of an example. Let us analyze the noun phrase problem statement:

MODERATELY SEVERE, ACUTE,
PNEUMOCOCCAL ARTHRITIS OF THE
LEFT KNEE

which is a typical problem statement.

The lexical phase

Of course, the only information about a word that an analyzer has is given to it by the system user. We must therefore relate each input word or compound term we wish the analyzer to recognize to a standard

term (which may be the term itself) and a part of speech. In some cases the part of speech may be changed later when context dictates such a move. For example, a noun might be effectively changed to an adjective role as in the case of liver becoming an adjective in liver biopsy. The form of such a lexicon might be as follows:

TERM OR COM- POUND TERM	STANDARD TERM	PART OF SPEECH
CEREBRAL VASCULAR ACCIDENT	CVA	NOUN
CVA	CVA	NOUN
DIABETES	DIABETESM	NOUN
DIABETES MELLITUS	DIABETESM	NOUN
DIABETES INSIPIDUS	DIABETESI	NOUN
MODERATELY SEVERE	SEVERE	ADJ
STROKE	CVA	NOUN
LIVER BIOPSY	LIVERBX	NOUN

For our problem statement, we would also use the information that "ACUTE" was substituted for "ACUTE" as an adjective, "PNEUMOCOCCUS" was substituted for "PNEUMOCOCCAL" as an adjective and "ARTHRITIS" was substituted for "ARTHRITIS" as a noun. Common incorrect spellings could be placed in the lexicon as well.

Syntactical phase

In this phase, the head term is separated out and the rest of the terms become adjectives. For example, the prepositional phrase is dealt with. In our sample problem "of the left knee" is such that "left knee" tells the location of the arthritis so "of the left knee" is converted to the role of an adjective. Once this is recognized, then "of the" is no longer required.

The extent of the adjectives modifying the noun within the prepositional phrase can be marked with a special symbol, say an ampersand. Some nouns used as adjectives will be taken care of in the lexical compressions (e.g., "liver" in "liver biopsy"). Others can be handled by assuming that the final noun in a string of nouns is really a noun and that the rest are adjectives (e.g., "disease" in "kidney disease"). The ambiguity which might arise because of confusion with regard to whether a term is a noun or a verb (e.g., "biopsy") does not arise since we are dealing exclusively with "simple" noun phrases.

Normalized text

The output of the lexical and syntactical phases combined is called "normalized text." It forms the input to the semantic parser. We are using as an example for analysis the problem statement:

MODERATELY SEVERE, ACUTE,
PNEUMOCOCCAL ARTHRITIS OF THE
LEFT KNEE

Our example would have the normalized text:

SEVERE,ACUTE,PNEUMOCOCCUS,
&LEFT,KNEE, ARTHRITIS

Tree directory

While it is possible to get some reduction to identical meaning by use of the lexicon alone, the power of modification and selection resides in the semantic parse itself. The elements of the semantic parse are the entries in the tree directory. Starting with the head term and its terms on the next node, the tree can be constructed. Two types of nodes are available. Both represent "term or terms on the next node." The first is an inclusive node in which all the terms are used. The second is a selector node in which only one of the possible choices is selected. One modifier being chosen is equivalent to a subset being chosen since the one selected can be defined to generate the others required. For display purposes, an asterisk in front of the members of a set of "terms on the next node" will denote that only one of those is to be selected. If one of the selections in turn has no entry in the tree directory, it is assumed to be terminal and "null" as the following step is automatically supplied. The form of the tree directory is as follows:

TREE DIRECTORY TERMS	TERMS ON NEXT NODE
ARTHRITIS	JOINT INFLAMMATION
BACTERIAL	*GONOCOCCUS *MENINGOCOCCUS *PNEUMOCOCCUS
DEGREE	*MILD *MODERATE *SEVERE

DURATION	*ACUTE *SUBACUTE *CHRONIC	DURATION = ACUTE = NULL =
ETIOLOGY	*INFECTIOUS *OSTEO *RHEUMATOID	ETIOLOGY = INFECTIOUS = BACTERIAL = PNEUMOCOCCUS = NULL =
INFECTIOUS	*BACTERIAL *VIRAL	LOCATION = JOINT =
INFLAMMATION	DEGREE DURATION ETIOLOGY LOCATION	JOINTNAME = KNEE = NULL =
JOINT	JOINTNAME SIDE	SIDE = LEFT = NULL =
JOINTNAME	*SHOULDER *FINGER *HIP *KNEE *TOE *POLY	
LOCATION	*JOINT *CAVITY *ORGAN	
SIDE	*LEFT *RIGHT *BOTH	

“JOINT” and “KNEE” are duplicated because if the problem had been stated “MODERATELY SEVERE, ACUTE PNEUMOCOCCAL INFLAMMATION OF THE LEFT KNEE,” we would need some way to indicate location (except see section on IMPLIED RELATIONS, below). The only result not thus far explained is how the terms “INFECTIOUS” and “BACTERIAL” were produced considering that they were not present in the original problem statement. This facility is taken up in the following section.

Implied relations

Within a given context, more information is present in terms of meaning than is given by the terms actually comprising a given statement. In a medical context, for example, the term “pneumococcal” calls to mind that the pneumococcus is a bacterial agent and therefore the process involved is an infectious one. Thus the fact that pneumococcal implies bacterial and bacterial implies infectious can be and was used to good advantage in producing the above sparse.

The logical relations, of course, are not always as simple as pure implication and for some cases one can see the practical extension of these operations to cover other cases. For example (Diabetes AND Pancreas) might imply “Endocrine.” The present system deals with pure implication, and the binary logical operators, AND and OR. The OR is an inclusive OR.

We note parenthetically that there is redundancy present in the implied relations. This really is not a bad situation except in the information theoretical transmission sense. We have two choices, first to put in implied members of the tree or second to cull out all such members which are otherwise implied. I have

The sparse

The skeletal form of the sparse is solely determined by the head term. Aside from that head term, the only role played by the normalized text is to supply selector node decisions. The tree produced by the sparse from the sample problem statement with reference to the above tree directory is as follows:

```

ARTHRTIS =
  JOINT =
    JOINTNAME =
      KNEE =
        NULL =
      SIDE =
        LEFT =
          NULL =
        INFLAMMATION =
          DEGREE =
            SEVERE =
              NULL =

```

chosen the first course of action even though it occupies more space, since it gives us the most consistent body of information to use for the similarity determination. This is a desirable approach to allow the greatest degree of symmetry since it exhibits linkages which might otherwise be lost. This "expansive" approach might well be criticized on the basis that somehow the "essence" of meaning should be the smallest set of words possible, but this is not one of my present goals. It appears, in fact, that the only method by which one can demonstrate the meaning of an item is to attach all the relevant sememic tags to it. Since words are often quite rich in meaning, we would expect to have a number of such tags occur and in fact this is part of the measure of "goodness."

The end-sparse

Once the sparse has been constructed, we have utilized the power of the hierarchical structure and now can transform the sparse to a form that is convenient for the purpose of information retrieval. A very convenient format is that of the linear descriptor string. Each data item is enclosed between virgules with the descriptor for that item to its right. This is an "attribute-value" approach. Such a record can be logically searched with the SEARCH program⁷ or similar systems. The sparse placed in the linear descriptor form is called an "end-sparse." Note that one item's descriptor can be another descriptor's item. For our sample problem we obtain:

```
/NULL/LEFT/SIDE/NULL/KNEE/JOINT-
NAME/JOINT/LOCATION/NULL/PNEUMO-
COCCUS/BACTERIAL/INFECTIOUS/
ETIOLOGY/NULL/ACUTE/DURATION/NULL/
SEVERE/DEGREE/INFLAMMATION/NULL/
LEFT/SIDE/NULL/KNEE/JOINTNAME/
JOINT/ARTHRITIS/
```

The end-sparse for the problem statement

"MODERATELY SEVERE, ACUTE, PNEUMOCOCCAL INFLAMMATION OF THE LEFT KNEE" would be the same except that "/ARTHRITIS/" (which is redundant information anyhow) would not occur. If one states in the implied relations that "joint AND inflammation IMPLIES arthritis," the head term "inflammation" will be replaced by the higher level term "arthritis." In any case, since the basic elements (those resulting from "joint" and "inflammation") do occur, we have transformed a statement said in different words than the previous one but with the similar meaning into a "similar" form. It would be

possible to remove redundancies and thus shorten the end-spares.

MEANINGEX IMPLEMENTATION

MEANINGEX is a language for "extracting meaning" from medical text which consists of problem statements. The language MEANINGEX runs interpretively on the CDC 3300 computer and has been implemented in the assembly language COMPASS. The current version is batch process only. The instructions are *INPUT LEXICON, *IMPLICATIONS, *TREE DIRECTORY, *DUMP LEXICON, *DUMP IMPLY LIST, *DUMP TREE DIRECTORY, and *SPARSE. Implications are described in Polish postfix notation. An on-line interactive system would be valuable since spelling, format and implication problems could be resolved in a conversational manner. Many of the items could be internally coded so the storage requirements for the spares could be pared down dramatically. In the present version where space was not a major factor, a design decision was made to maintain everything in actual text.

EXAMPLE OF MEANINGEX ANALYSIS

Two spares from a typical run using the MEANINGEX interpreter appear in Figure 1.

Comparison of the first and second problem statements illustrates the use of the implication facility to displace a head term. Implication is stated in Polish postfix notation with = standing for implication, * for an inclusive OR, and & for AND. In the second sparse, the implication "JOINT,INFLAMMATION,& ARTHRITIS,=" is used to displace the head term "INFLAMMATION" by "ARTHRITIS." Thus the spares:

```
MODERATELY SEVERE, ACUTE,
PNEUMOCOCCAL ARTHRITIS OF THE
LEFT KNEE
```

and

```
SEVERE, ACUTE PNEUMOCOCCAL
INFLAMMATION OF THE LEFT KNEE
```

are transformed to an identical form which was the object of the semantic analysis. Thus "LOCATION" as a tree directory nextnode for inflammation is redundant in this case. Note that in the first two spares, the information that the condition is serious (from the implication "SUBACUTE,ACUTE,*,SEVERE,&,SERIOUS,=") and that a bacterial and there-

```

*SPARSE
.1075938
MODERATELY SEVERE, ACUTE, PNEUMOCOCCAL ARTHRITIS OF THE LEFT KNEE *
SEVERE, ACUTE, PNEUMOCOCCUS, &LEFT, KNEE, ARTHRITIS,
ARTHRTIIS=
JOINT=
  JOINTNAME=
    KNEE=
      NULL=
  SIDE=
    LEFT=
      NULL=
INFLAMMATION=
  DEGREE=
    SEVERE=
      NULL=
  DURATION=
    ACUTE=
      NULL=
  STATUS=
    SERIOUS=
      NULL=
  ETIOLOGY=
    INFECTIOUS=
      BACTERIAL=
        PNEUMOCOCCUS=
          NULL=
  LOCATION=
    JOINT=
      JOINTNAME=
        KNEE=
          NULL=
      SIDE=
        LEFT=
          NULL=

.1075938/NULL/LEFT/SIDE/NULL/KNEE/JOINTNAME/JOINT/LOCATION/NULL/PNEUMOCOCCUS/BACTERIAL/INFECTIOUS/ETIOLOGY/NULL/SERIOUS/
STATUS/NULL/ACUTE/DURATION/NULL/SEVERE/DEGREE/INFLAMMATION/NULL/LEFT/SIDE/NULL/KNEE/JOINTNAME/JOINT/ARTHRTIIS/

.0935487
SEVERE, ACUTE PNEUMOCOCCAL INFLAMMATION OF THE LEFT KNEE *
SEVERE, ACUTE, PNEUMOCOCCUS, &LEFT, KNEE, INFLAMMATION,
SEVERE, ACUTE, PNEUMOCOCCUS, &LEFT, KNEE, ARTHRITIS,
ARTHRTIIS=
JOINT=
  JOINTNAME=
    KNEE=
      NULL=
  SIDE=
    LEFT=
      NULL=
INFLAMMATION=
  DEGREE=
    SEVERE=
      NULL=
  DURATION=
    ACUTE=
      NULL=
  STATUS=
    SERIOUS=
      NULL=
  ETIOLOGY=
    INFECTIOUS=
      BACTERIAL=
        PNEUMOCOCCUS=
          NULL=
  LOCATION=
    JOINT=
      JOINTNAME=
        KNEE=
          NULL=
      SIDE=
        LEFT=
          NULL=

.0935487/NULL/LEFT/SIDE/NULL/KNEE/JOINTNAME/JOINT/LOCATION/NULL/PNEUMOCOCCUS/BACTERIAL/INFECTIOUS/ETIOLOGY/NULL/SERIOUS/
STATUS/NULL/ACUTE/DURATION/NULL/SEVERE/DEGREE/INFLAMMATION/NULL/LEFT/SIDE/NULL/KNEE/JOINTNAME/JOINT/ARTHRTIIS/

```

Figure 1—Sparses from a typical MEANINGEX run

fore infectious process is involved (from the implications "PNEUMOCOCCAL, INFECTIOUS, =" and "VIRAL, PPLO, *, MYCOBACTERIUM, *, BACTERIAL, *, INFECTIOUS, =") does not occur directly in the problem statements, but is added because the

system "understands" through its use of stored semantic information.

A striking feature of the system is the versatility of a relatively small number of tree directory entries. The trees are built modularly so frequently used

information such as "INFLAMMATION," "ORGAN" and "BACTERIAL" need only appear once and yet contribute a great deal of semantic information to many trees.

SEARCH REQUEST TRANSFORMATION

We have only talked about transforming the original problem statement. This leaves the burden of the appropriate choice of relatively "low level" descriptors to the user who is formulating the search requests. The MEANINGEX approach could be used to automatically transform search requests using the identical lexicon, tree directory, etc. used to sparse the problem statements.

MEANINGEX vs. SEMANTIC ANALYZER REQUIREMENTS

Five requirements for a semantic analyzer were outlined above. The ability for MEANINGEX to fulfill these criteria is dealt with in the following discussion.

Semantic transformation

The first requirement was that statements which are given in different words but are recognized as meaning the same thing in a given contextual communication are reduced to an identical form. This is exactly what MEANINGEX can do very well. The transformation to an identical form requires that the implications be set up so that "higher level" head terms can displace lower level ones. Another part of this requirement is actually that similar statements be transformed to a similar form. This will occur to a great extent whether head term displacement is employed or not. In the majority of instances of information retrieval, one is interested in selecting out records with certain descriptors or descriptor-value pairs, and not getting an exact match. For example one is more likely to be interested in all those records with infectious arthritis of some type, than be concerned with the degree of severity of the process.

Selection of sense of meaning

The selection of sense of meaning goes back to the selection of the appropriate reading of possible derivations for a term present in a lexicon of the type suggested

in the Katz, Fodor and Postal model which was discussed above. This capability in MEANINGEX is embodied in the selector node. Selections are automatically made based on the information present in the rest of the problem statement being sparsed. Indeed this use of contextual information to determine sense of meaning is the only way this process can be done.

Resolution of ambiguity

Ambiguity is a problem for MEANINGEX. Depending on the situation, MEANINGEX will choose one meaning and lose the other possibility. The capability for handling the conjunction "and" has not yet been put in the system. This term is a particularly likely source for ambiguity. Take, for example, the problem statement:

SEVERE MYOCARDIAL INFARCTION AND PULMONARY EDEMA

which might well appear. The ambiguity as to whether "severe" applies to both "myocardial infarction" and "pulmonary edema" or just to "myocardial infarction" alone can be only "resolved" in the sense that a convention is stated. In the case of MEANINGEX, this could mean breaking up the problem statement into two parts with a decision made as to the range of "severe." One could leave "pulmonary edema" attached to the "myocardial infarction" as well, so it perhaps could be picked up as an associated condition. This is not really necessary, however, since the term "pulmonary edema" will end up with a sparse under that particular patient's record number anyhow.

Deletion of contradiction

At this point in time, the ability of MEANINGEX to delete contradiction resides in the fact that once a head term has been chosen, a skeletal tree is determined (albeit with a number of selector node choices possible for modification of meaning). Terms that do not fit into the possibilities inherent in that skeletal sparse are left over. This tends to maintain an internally consistent view of the problem and will often delete contradictions. It is true that some correct items might be deleted because of incompleteness in the tree directory. Examination of such deleted items will, however, point to the corrections to be made.

Specificity problem

Since MEANINGEX allows arbitrary modification to any depth, one can be quite specific if appropriate tree directory entries are used. The way the system is set up, it is possible to decide that a terminal requires further modification and just add the related tree directory entry. It might be desirable, for example, to further break down a particular kind of bacteria. Another type of specificity would be provided if one had quantitative capabilities.

MEANINGEX AND FREE TEXT

A consideration is the use of MEANINGEX for the encoding of meaning in free text. If a period were replaced by "*", the system could analyze sentences rather than noun phrases. In the most trivial application, verbs would be ignored. At another level, just the determination would be made as to whether the present or past tense was involved. An adjective could then be produced and attached to the sparse tree. For example, if in the lexicon we had the entries:

IS, PRESENT, ADJ,
WAS, PAST, ADJ,
WILL BE, FUTURE, ADJ,

we could have in the tree directory:

INFLAMMATION = /DEGREE/DURATION/
ETIOLOGY/CHRONOLOGY/

and

CHRONOLOGY = */PRESENT/PAST/FUTURE/

Another lexical item defined as /PAST/ would be "HISTORY OF" defined as an adjective. Higher level use would require a more detailed syntactic analysis.

CONCLUSION

MEANINGEX is perhaps the first system for extracting meaning by describing terms using arbitrary modification. It uses a functional rather than a philosophical approach, although the basic mechanism of selection is grounded in linguistic theory of the past decade. The basic principle is that we want to take noun phrases and transform them such that statements that are said in different words but which humans would recognize mean the same thing are transformed to an essentially identical product. Although the examples involve medical text, the existing mechanics are not restricted to it.

REFERENCES

- 1 L WEED
Medical records that guide and teach
New England Journal of Medicine Vol 278 No 11
pp 593-600 No 12 pp 652-657 1968
- 2 L WEED
Medical records, medical education, and patient care
Case Western Reserve University Cleveland Ohio 1969
- 3 J J KATZ P M POSTAL
An integrated theory of linguistic descriptions
MIT Press Cambridge Massachusetts 1964
- 4 J J KATZ J A FODOR
The structure of a semantic theory
pp 479-518 in *The structure of language* (Fodor and Katz
editors) Prentice-Hall Englewood Cliffs New Jersey 1963
- 5 AMERICAN MEDICAL ASSOCIATION
Standard nomenclature of diseases and operations
Blakiston Division McGraw-Hill Book Company
New York 1961
- 6 WORLD HEALTH ORGANIZATION
*Manual of the international statistical classification of
diseases, injuries and causes of death*
2 Volumes Geneva Switzerland 1957
- 7 U S PUBLIC HEALTH SERVICE
International classification of diseases, adapted
U S Department of Health Education and Welfare
Washington D C 1962
- 8 COMMITTEE ON NOMENCLATURE AND
CLASSIFICATION OF DISEASE OF THE
COLLEGE OF AMERICAN PATHOLOGISTS
Systematized nomenclature of pathology
College of American Pathology Chicago Illinois 1965
- 9 D A B LINDBERG
Electronic retrieval of clinical data
Journal of Medical Education Vol 40 No 8 pp 753-759 1965
- 10 B L GORDON
*Biomedical language and format for manual and
computer applications*
Diseases of the Chest Vol 53 No 1 pp 38-42 1968
- 11 B L GORDON (editor)
Current medical terminology
American Medical Association Chicago Illinois 1966
- 12 B G LAMSON BERNICE C GLINSKI
G S HAWTHORNE J C SOUTTER
W S RUSSELL
*Storage and retrieval of uncoded tissue pathology
diagnoses in the original English free-text form*
Proceedings of the 7th IBM Medical Symposium
Poughkeepsie New York 1965
- 13 B G LAMSON B DIMSDALE
A natural language retrieval system
Proceedings of the IEEE Vol 54 No 12 pp 1636-1640 1966
- 14 H JACOBS
A natural language retrieval system
Methods of Information in Medicine Vol 7 No 1
pp 8-16 1968
- 15 P SHAPIRO
ACORN—an automated coder of report narrative
Methods of Information in Medicine Vol 6 No 2
pp 153-162 1967
- 16 P A SHAPIRO I D BROSS R L PRIORE
B B ANDERSON
Information in natural language, a new approach

- Journal of the American Medical Association Vol 207
No 11 pp 2080-2084 1969
- 17 R F SIMMONS
Answering English questions by computer: a survey
Communications of the ACM Vol 8 No 1 pp 53-69 1965
- 18 R F SIMMONS
Natural language question-answering systems: 1969
Communications of the ACM Vol 13 No 1 pp 15-30 1970
- 19 J D BEATTIE
Natural language processing by computer
International Journal of Man-Machine Studies Vol 1
No 4 pp 311-329 1969
- 20 R F SIMMONS S KLEIN K L MCCONOLOGUE
Indexing and dependency logic for answering English questions
American Documentation Vol 15 No 2 pp 196-204 1964
- 21 V E GIULIANO
Comments (on reference 17)
Communications of the ACM Vol 8 No 1 pp 69-70 1965
- 22 R K LINDSAY
Inferential memory as the basis of machines which understand natural language
pp 217-233 in *Computers and thought* (Feigenbaum and Feldman editors) McGraw-Hill New York 1963
- 23 B F GREEN A K WOLF
C CHOMSKY K LAUGHERY
Baseball: an automatic question answerer
pp 207-216 in *Computers and thought* (Feigenbaum and Feldman editors) McGraw-Hill New York 1963
- 24 R A KIRSCH
Computer interpretation of English text and picture patterns
IEEE Transactions on Electronic Computers
Vol EC-13 No 4 pp 363-376 1964
- 25 R M SCHWARZ J F BURGER R F SIMMONS
A deductive question-answerer for natural language inference
Communications of the ACM Vol 13 No 3 pp 167-183 1970
- 26 B RAPHAEL
SIR: a computer program for semantic information retrieval
PhD Dissertation MIT 1964
- 27 B RAPHAEL
SIR: a computer program for semantic information retrieval
pp 33-145 in *Semantic information processing* (Minsky editor) MIT Press Cambridge Massachusetts 1968
- 28 D G BOBROW
Natural language input for a computer problem-solving system
PhD Dissertation MIT 1964
- 29 D G BOBROW
Natural language input for a computer problem-solving system
pp 146-226 in *Semantic information processing* (Minsky editor) MIT Press Cambridge Massachusetts 1968
- 30 K M COLBY H ENEA
Heuristic methods for computer understanding of natural language in context-restricted on-line dialogues
Mathematical Biosciences Vol 1 No 1 pp 1-25 1967
- 31 L TESLER H ENEA K M COLBY
A directed graph representation for computer simulation of belief systems
Mathematical Biosciences Vol 2 No 1 pp 19-40 1968
- 32 K M COLBY H ENEA
Machine utilization of the natural language word "good"
Mathematical Biosciences Vol 2 No 1 pp 159-163 1968
- 33 J WEIZENBAUM
ELIZA—a computer program for the study of natural language communication between man and machine
Communications of the ACM Vol 9 No 1 pp 36-45 1966
- 34 J WEIZENBAUM
Contextual understanding by computers
Communications of the ACM Vol 10 No 8 pp 474-480 1967
- 35 M R QUILLIAN
Semantic memory
PhD Dissertation Carnegie Institute of Technology
Pittsburgh Pennsylvania 1966
- 36 M R QUILLIAN
Semantic memory
pp 227-270 in *Semantic information processing* (Minsky editor) MIT Press Cambridge Massachusetts 1968
- 37 M R QUILLIAN
The teachable language comprehender: a simulation program and theory of language
Communications of the ACM Vol 12 No 8 pp 459-476 1969
- 38 D C DUNPHY P J STONE M S SMITH
The General Inquirer: further developments in a computer system for content analysis of verbal data in the social sciences
Behavioral Science Vol 10 No 4 pp 468-480 1965
- 39 P J STONE D C DUNPHY M S SMITH
D M OGILVIE
The General Inquirer
MIT Press Cambridge Massachusetts 1966
- 40 P J STONE D C DUNPHY M S SMITH
D M OGILVIE
User's manual for the General Inquirer
MIT Press Cambridge Massachusetts 1967
- 41 G SALTON
The evaluation of automatic retrieval procedures—selected test results using the SMART system
American Documentation Vol 16 No 3 pp 209-222 1965
- 42 G SALTON
A comparison between manual and automatic indexing methods
American Documentation Vol 20 No 1 1969
- 43 G SALTON
Automatic text analysis
Science Vol 168 No 3929 pp 335-343 1970
- 44 G SALTON M E LESK
Computer evaluation of indexing and text processing
Journal of the ACM Vol 15 No 1 pp 8-36 1968
- 45 M E LESK G SALTON
Interactive search and retrieval methods using automatic information displays
Technical report No 68-17 Department of Computer Science Cornell University Ithaca New York 1968
- 46 E IDE G SALTON
User-controlled file organization and search strategies
Proceedings of the American Society for Information Science Vol 6 pp 183-191 1969
- 47 R P RICH
Information handling
Proceedings of the 6th IBM Medical Symposium
Poughkeepsie New York pp 197-206 1964

Law enforcement communication and inquiry systems

by JOHN D. HODGES, JR.

Continental Information Systems
Santa Monica, California

The Law Enforcement and Criminal Justice Community is one that survives and becomes effective through the efficient use of information. People are wanted for criminal offenses, warrants are issued, vehicles and property are stolen or recovered, associated criminal activities and known or suspected offenders must be traced—a mass of data that must be collected, verified, stored and disseminated to the appropriate people in a timely and reliable manner. Elements that compound the problem are geographic dispersion, availability of rapid transportation means and the mobility of the criminal element, the criticality of the data to the officer on the street and the sensitivity of the data in the personal privacy realm. This information need of Law Enforcement has been approached by the various national, state and local agencies as a communication problem. Their original solution was a series of inter-connected teletype systems that transferred wanted notices, requests for data, etc., between and within agencies. But as society has become larger and more complicated, these agencies have had to enhance their information transfer and storage capabilities.

In the last two to three years more than twenty law enforcement computerized communication and inquiry systems have been installed or are in the process of implementation. This paper presents the basic requirements for these systems as reflected by the specifications developed by the user agencies and the computer applications necessary to satisfy the specifications.

SYSTEM REQUIREMENTS

In order to establish what the law enforcement agencies have defined as their system needs, twelve (12) Request for Proposals from user groups have been analyzed and a set of requirements established. This analysis is presented in Table I, where the associated requirement of each requesting agency is specified. As

can be readily seen, these systems are both communication and inquiry systems with a background batch processing requirement. They require a range of inquiry subsystems and computer-to-computer interfaces. All types of terminals and communication circuits must be supported, in a variety of combinations and volumes. The systems must all operate on a 24-hour 7-day week basis and the majority require some type of system back-up. They have a varied schedule of response times, but most expect a ten (10) second or less terminal-to-terminal response time.

To indicate the general requirements of these systems, the description of requested systems by three agencies will be presented.

The Police Department of The Metropolitan Government of Nashville and Davidson County defined the general requirements for their real-time law enforcement information system as follows:

The electronic data processing equipment to be installed for the Metro Nashville Police Department will serve a dual purpose function for the law enforcement community in the Middle Tennessee area—data communications and information retrieval.

The objective is to acquire a modularly expandable hardware configuration capable of processing increasing workloads. The installation of this equipment is intended to provide the basis for automating the procedures and capabilities of data communications in and around Nashville, Tennessee. This will be accomplished through the ultimate employment of up to 100 remote telecommunication terminals connected on-line to the central system configuration located in Nashville, Tennessee.

The system will be a multi-purpose system and must simultaneously perform several tasks. The tasks are to be performed under priority rules with automatic transition from one priority level to

TABLE I—Law Enforcement Communication and Inquiry System Requirements

PAGE 1

Requirement	Colo.	Fla.	Nash-ville	N.J.	N.Y.	N.C.	Mo.	Okla.	Ontario	Ore.	Tex.	Wash.
Functions												
Message Switching	X	X	X	X	X	X	X	X	X	X	X	X
On-Line Information Storage, Retrieval and Dissemination	X	X	X	X	X	X	X	X	X	X		
Batch Data Processing	X	X	X	X	X	X	X	X	X	X		
Software Applications												
Master Name Index	X	X	X	X		X						
Vehicle	X	X	X	X	X	X	X	X		X		
Person	X	X	X	X		X	X	X	X	X		
Property	X	X	X	X	X	X	X	X	X	X		
Want-Warrant	X	X	X	X		X	X	X	X	X		
File Management	X	X	X	X	X	X	X	X	X	X		
NCIC Interface	X	X	X	X	X	X	X	X		X	X	X
Message Switching	X	X	X	X	X	X	X	X	X	X	X	X
Computer Interfaces	X	X	X	X	X	X	X		X	X	X	X
Terminal Types												
Teletype	X			X	X		X	X	X	X	X	X
Other Hard Copy	X	X	X	X	X	X	X			X		
CRT Devices	X	X	X	X	X	X		X		X		
Facsimile	X				X					X		
Number of Terminals	38-200	280	25-100	235-400	365-500	150-300	15-30	65	110	80-100	250	110
Type of Circuits												
Low Speed, Teletype Grade	X	X	X	X	X	X		X	X	X	X	X
Low Speed, Voice Grade	X	X	X	X	X	X	X	X	X	X	X	X
High Speed, Asynchronous	X	X	X	X	X	X				X	X	X
High Speed, Synchronous	X	X	X	X	X	X				X	X	X
Simplex				X							X	
Half Duplex	X	X	X	X	X	X	X	X	X	X	X	X
Full Duplex	X				X					X	X	
Number of Circuits	15	17-78	6-18	14-32	146	13-42	15-30	7-16	110	12	15	33-60
Number of Computer Interfaces	4-5	9	2	8	12	4	2	1	2	3	3	5
24 Hour Per Day-7 Day Per Week Operation	X	X	X	X	X	X	X	X	X	X	X	X
Back-Up System							No	No	No		No	
Duplex Processors				X		X						X
Redundant					X							
Message Switching	X	X								X		
Miscellaneous			X							X		

TABLE I (cont'd)—Law Enforcement Communication and Inquiry System Requirements

PAGE 2

Requirement	Colo.	Fla.	Nash-ville	N.J.	N.Y.	N.C.	Mo.	Okla.	Ontario	Ore.	Tex.	Wash.
Files											None	None
Master Name Index	X	X	X	X		X			X	X		
Wanted Persons	X	X	X	X		X	X	X	X	X		
Criminal History		X	X	X		X		X	X			
Stolen Vehicles	X	X	X	X	X	X	X	X	X	X		
Stolen Guns	X			X	X			X		X		
Stolen Articles (Property)	X	X	X	X	X	X	X	X	X	X		
Drivers Records								X				
Vehicle Registration								X				
Terminal Response Time*												
90% of Transactions	10s									10s	5s	
Inquiries					5s							
Messages					5m							
99% of Transactions	60s	30s	10s			10s			30s	60s		
Inquiries				10s			20s	20s				
Update				3m								
Messages				5m			60s	60s				
Maintenance Response	1hr		1 hr		½ hr	1 hr				1 hr		
Standard of Performance												
System Acceptance	95%		95%	**	95%	95%						93%
System Operation												93%
Demonstration***												
Hardware & Gen. Sup. Software	Opt			Opt	Opt			Opt	Opt	Opt		Opt
Application Software	Opt			Opt	Opt					Opt		
Benchmark	Opt	Opt	Opt	Opt		Opt	Req	Opt		Opt		
Multi-Programming	X		X		X	X	X	X		X		
Storage Protection	X				X					X	X	
Dynamic Loading & Linking	X		X			X				X	X	
Language Processors												
Assembler	Req	Req	Req	Req	Req	Req	Req	Req	Req	Req	Req	
COBOL	Req	Req	Req	Req	Req	Req	Req	Req	Req	Req		
FORTRAIN	Opt	Req	Req	Req	Req	Req	Req	Req	Req	Req	Opt	
RPG	Opt									Opt		
Transaction Processing—Peak Hour												
Inquiry and Update		8100	395	1980	330-550		124	250		1000		
Messages		1000		360	1020-1500		60				4500	3400

* s = seconds

m = minutes

** Mutually agreed to Acceptance Test

*** Opt = Optional

Req = Required

another in a manner calculated to optimize the total system utilization.

The tasks to be performed by the system fall into three main categories:

- (1) Information storage/retrieval/dissemination
- (2) Message switching
- (3) Background batch processing

... The proposed configuration must function with or exhibit the following characteristics:

- a. *General Purpose*: i.e., not oriented to scientific or special applications, except as in b. below.
- b. *Communications Oriented*: capable of supporting up to 100 remote telecommunication terminal devices for on-line transmission.
- c. *Multi-Programming Capability*: capability of concurrently processing the tasks listed above.
- d. *High Performance*: must take advantage of current technical advances in computer design, systems organization and programming technology to insure a high level of efficiency and processing throughout.
- e. *Capable of Modular Expansion*: must provide for hardware and basic operating systems software compatibility to facilitate installation upgrading and expansion. Expansion capability for reasonable system growth in the next five years in both input, storage and processing without major hardware change or reprogramming is a requirement.

The Request for Proposal for Colorado's Automated Law Enforcement System documented the system concept and capabilities in the following manner:

The Colorado Law Enforcement Information System (CLEIS) will be an on-line, computer based, communications, information retrieval, and centralized records system, serving Colorado law enforcement agencies. The initial number of terminals are estimated at thirty-eight (38); eventually a maximum number of two hundred (200) will be on-line when the total criminal justice system is fully operational. It is a requirement that the current thirty-eight (38) ASR 28 Colorado Law Enforcement Teletype System terminals be supported in the first part of Phase I. In addition to complete message switching capability between agencies (CLETS and NLETS) and file inquiry and updating in the application areas, the system will provide automatic interfaces with other local, state, and national law enforcement systems as

follows:

- National Law Enforcement Teletype System (NLETS)—Phoenix, Arizona.
- National Crime Information Center (NCIC), Washington, D. C.
- Department of Revenue—Motor Vehicle Registrations and Drivers Licenses inquiry—Capitol Complex, Denver.
- Denver Crime Information Center (DCIC), Denver, Colorado.
- Other systems such as Project SEARCH as the need for such interfaces arise.

These automatic interfaces will give all CLEIS users on-line access to the automated Driver's License and Vehicle Registration files at the Department of Revenue, DCIC information which will not be duplicated within CLEIS, and to the general crime information files at NCIC, as well as give users on DCIC access to state and national files through a common terminal.

The system proposed must perform three primary tasks (message switching, on-line file inquiry and update, and batch data processing) concurrently. Message switching and on-line file inquiry and updating will be operational twenty-four hours a day, seven days a week. Batch processing will be available as required to support the on-line tasks. It is a requirement that the ability to automatically transmit messages from point to point not be affected by a failure in the central files or processor(s).

The system requirements of the State of New Jersey for the Division of State Police Statewide Communications Information System were set forth in the following description:

This communications system is designed to support the law enforcement agencies of the State of New Jersey. A digital computer complex consisting of two communications-oriented computer processing units (CPUs) and their related peripheral equipment, will be connected by telecommunication lines in an array of communication terminal devices remotely located at State Police facilities, at facilities of New Jersey county and municipal police agencies and at Motor Vehicle offices and other locations.

Duplexed CPUs and switchable peripherals will provide communications processing backup for the system. The primary communications processor

will handle the State Police digital data communications, information storage and retrieval, and related background batch processing. The secondary or back-up communication processor will normally handle the other communications needs of the Department of Law and Public Safety. In a backup situation either processor must be capable of performing all communications tasks.

The communications tasks related to the Division of Motor Vehicle consist of concentrating messages from terminals and forwarding them to the Division of Motor Vehicles data processors for updating and inquiry handling. Responses will be transmitted back to the originating terminals.

The communications processors are also linked to other communication systems outside the state with which information is exchanged; for example, the National Crime Information Center (NCIC), located in Washington, D. C.

Two types of data messages are carried on the communications system. The first type is those messages originating at the terminal for transmission to one or more other terminals. Administrative messages, bulletins and alarms fall into this category. The identity of each of the receiving terminals is incorporated into the message header when it is composed.

The second general type of message is that associated with inquiries. An inquiry message composed at a terminal, is used by the computer complex to elicit information from one or more files maintained by the system. Information so derived then is used by the computer complex to compose a response message which is routed back to the inquiring terminal. Certain inquiries, depending on their content, will be reformatted and routed to either DMV data processor or to the NCIC system. The receiving system formulates the response to the inquiry and routes this response back to the communications processor, where it is again routed to the inquiring terminal.

The inquiry-handling sub-section involves entry, update, cancellation and inquiry messages. To introduce a new entry into a data file requires that an entry message be composed at a terminal. Upon receipt of this message, the computer complex creates new records in the pertinent file. Some entry messages with the appropriate content will be forwarded to NCIC for inclusion in their files.

New information on an existent record is introduced into the system by an update message composed at a terminal. The computer complex uses this message to modify or update the content

of a record. Related records in different files may be modified by an update message.

Deletion of a particular entry in a file (or related entries in different files) is done by means of a cancel message, which codes the record for elimination. However, the actual act of purging the entry is performed later, with a time-lapse which depends upon the retention period of the file involved.

Receipt of an entry, update or cancel message by the computer complex initiates an acknowledgment message being transmitted back to the originating terminal by the complex. However, if the received message is found to be in error, a message describing the error condition is passed back instead.

The system will provide for keyboard terminal devices equipped with a cathode-ray tube (CRT) device in addition to a printing device. This allows messages to be composed and viewed on the CRT as well as being printed out in hard copy form. Terminals of this sort are buffered and are connected to the computer complex by means of voiceband, half-duplex lines. Each of these terminals may have its own point-to-point line connection with the computer complex, or a given line may be shared by a number of such terminals. However, this shared arrangement will prevent one terminal from making use of the line if it is being used by another terminal. The full configuration could include 75 or more such terminals, located at State Police and municipal locations.

The full configuration also can provide for up to 190 Receive-Only (RO) terminals. These devices can receive messages from the computer complex and reproduce these in hardcopy but do not have the capability of entering (composing) messages. These terminals may share some 25 narrowband, multi-drop lines, with five to ten terminals sharing a line.

The following sections discuss the Communications, Inquiry and File Management capabilities to meet these system requirements.

COMMUNICATIONS

The communications capabilities encompass the routing of all messages and control of all terminal stations within a law enforcement network.

Each terminal associated with the system is assigned capabilities as to the types of messages the terminal is permitted to send and receive: e.g., QUERY, UPDATE, SWITCHED, FUNCTIONAL, CONTROL; and the

level of control that the terminal is permitted to exercise over message processing operations and the operations of other terminals: e.g., STANDARD, PRIVILEGED, MASTER.

Each input message is checked for validity prior to assignment to one of the processing levels. The agency must specify the processing order of the validated SWITCHED, FUNCTIONAL AND CONTROL messages and the terminals that are permitted to monitor and even intercept the traffic routed to other terminals in the network. Within the system query messages maintain the highest processing priority, with update messages usually holding the second priority.

Extensive fail-safe features must be provided to minimize the disruption of operations and enable an orderly changeover to a backup mode. The restart procedures should be highly automated and require a minimum of assistance from computer operators and include various categories of restart procedures.

In order to enable efficient reallocation of the system resources and provide statistics concerning terminal utilization, the system should automatically record and compile a number of measures of system activity, and present them in a form convenient for analysis.

Message types

A SWITCHED message contains unformatted text which is to be forwarded to the addressees specified. Such messages describe operational or administrative matters of interest at particular command levels or areas of activity within the Agency. SWITCHED messages may be directed within the immediate User network exclusively, or to remote systems such as the National Crime Information Center (NCIC). Interfaces to other systems whose central communications switching computer is connected may be handled as SWITCHED messages.

A QUERY message is a request for law enforcement file data and is directed by the originator to one of the application subsystems (such as Vehicle or Persons) rather than directly to another terminal. A QUERY message contains file search identifiers which a particular subsystem uses to locate records of interest within its associated files. Subsystems formulate and transmit messages containing the requested file data to the QUERY originators.

An UPDATE message contains the data required to modify the contents of an on-line data file: for example, add a record, modify elements in a record or delete the record. The subsystem returns a message to the originating terminal acknowledging the update.

A FUNCTIONAL message provides a terminal

operator with terminal oriented services. These services may include: deferring message output or checking terminal status.

The CONTROL message permits personnel at qualified terminals to alter basic message processing procedures within the System, such as to activate a routing list or to deactivate a communication line.

Level of control

A terminal operator can request message processing services or modify the message processing procedure depending upon the level of control assigned to his terminal. The three levels of control usually provided correspond to command levels within the agency organization: Standard Control, exercised at subordinate levels; Privileged Control, which is available at supervisory levels within the agency organization and within special headquarters and divisions; and Master Control, restricted to those terminals designated as having maximum command and technical responsibility for System operations.

Switched message processing

The message switching function must include the full range of capabilities normally associated with a generalized, selective store-and-forward system. It provides for the routing of messages between all terminals on the communications network including selecting multipoint and full broadcast capabilities. The message switching module should consist of numerous routines for routing determination, access and routing validation, routing redirection and message deferral.

The originator of a switched message should be able to specify routing destinations using any combination of at least three methods.

Single Station—Individual station identifiers referenced in a switched message header qualify particular stations as addressees.

List—Each station in the system can be assigned as a member of one or more routing lists. Reference to a routing list in a switched message header routes the message to all members of the list.

Geographic—The originator specifies a radius in miles, or other agency defined units, which describes an area with the originating terminal at the center. All stations in the described area qualify as addressees.

Multiple routing lists and multiple single stations

may be specified. In addition, combinations of geographic areas, routing lists and single stations may be specified in a single message. If a station appears on more than one routing list specified by the originating station, only a single copy of the message is to be forwarded.

The routing data associated with a SWITCHED message is modified when a non-addressee terminal is monitoring the output or input from terminals, or when due to terminal malfunction or other reason any of the addressee terminals are unable to accept output and other terminals within the network have been designated as alternate addressees. Duplicate copies of output messages are transmitted to those terminals which are monitoring the traffic of SWITCHED message addressees.

A file of undeliverable messages is established for each station which has requested that its output be held by the system and for each station which is determined as not being able to receive. Messages from these files may be removed upon receipt of FUNCTIONAL messages from the terminal indicating that output can once again be accepted; or printed on the line printer at the computer installation and forwarded to the addressee in hard copy form.

Functional message processing

FUNCTIONAL messages allow a station operator to request terminal oriented services directly from the Communications Module. FUNCTIONAL messages are subclassified into three groups according to the level of the stations authorized to invoke them; STANDARD, PRIVILEGED and MASTER.

Standard terminal oriented FUNCTIONAL messages that can be exercised by any terminals in the network include obtaining the current status of the terminal, operator log-in and placing the terminal in the test/training mode. While the terminal is in test-training mode, the operator may originate messages for validation by the system; however, switched messages are not forwarded to the addressee stations. The terminal status information includes the terminals designation and alternate; operator's identifier; current message number and the number of output messages currently queued.

FUNCTIONAL messages may be used to control the disposition of the output currently being directed to the terminal. For example, all output messages may be held until further notice. This capability is useful when priority input activity is expected or the terminal is to be placed off-line for some reason (for example, a paper or ribbon change).

The terminal operator may also obtain copies of previously transmitted messages. Either a single message may be obtained by specifying the appropriate message number, or multiple messages between specified message numbers may be retransmitted. In addition, the operator may request a retransmission of messages that were transmitted between specified times. Copies of previously transmitted messages are usually maintained by an on-line wrap around file. Only messages currently resident in the file may be retrieved on-line with other messages maintained on tape.

In addition to the FUNCTIONAL messages described above pertaining to his own terminal, the operator of a PRIVILEGED or MASTER terminal can affect the operation of other terminals. For example: hold the output to another terminal; direct the discarding of output of another terminal or have the output printed at the central site. He may also obtain the current status of other terminals in the network.

The operator of a MASTER or PRIVILEGED class terminal may selectively monitor traffic directed to other terminals. All traffic, or all traffic of a specified message type (SWITCHED, FUNCTIONAL or CONTROL), input only or output only traffic may be monitored. The monitoring of transmissions should not affect the routing of the messages, nor should the monitored terminal be aware of the monitoring.

Transmissions directed to, or originated from, another terminal may be intercepted. This intercept capability is essentially a monitoring function followed by a decision to either approve and forward the intercepted traffic; disapprove and cancel it; modify routing or priority and forward; or, hold for later action. The originator of an intercepted message is notified of its interception and disposition.

In addition to the messages described above, MASTER class terminals may also Monitor and Intercept line traffic and check line status. The monitor and intercept of line traffic is similar to the corresponding functions for single terminals. In this case, however, traffic on the entire line is considered.

Control message processing

CONTROL messages are used to alter the basic organization and operation of the communications network. The use of CONTROL messages is thus restricted to MASTER class terminals.

CONTROL messages can be used to activate or deactivate terminals or communications lines. When a line or terminal is deactivated, the associated polling and line control operations are suspended. In addition, all output directed to the line or terminal is held and transmitted when the line is reactivated.

The operator of a MASTER class terminal should have the ability to alter routing lists. Lists may be deactivated or terminals dynamically added to or deleted from the list. New routing lists may also be established.

In order to prevent abuse or unauthorized use of the functional monitor and intercept capabilities described above, CONTROL messages are provided to designate the terminals authorized to monitor or intercept traffic. Following this operation, the specified terminal may use the appropriate FUNCTIONAL message to initiate the monitor or intercept mode.

CONTROL messages are also provided to specify alternate terminals authorized to receive traffic in the event the primary terminal is not operational. Alternate terminals can assume the status, routing lists and other characteristics of the primary terminal or may be restricted to only receiving traffic. In this latter case, the alternate terminal maintains its own status, routing lists and other characteristics.

Message validation, assignment and priority

The construction of each incoming message is validated for format and content. Messages that are determined to be valid should be acknowledged with an appropriate response to the originating operator. The acknowledgment contains the message number assigned to the message. Messages that are found to be incorrect are returned to the originating operator with a diagnostic message to assist him in the correction of the message. The invalid message is discarded with no message number assigned.

Valid messages are assigned to a processing level (queue). SWITCHED messages may be submitted with Emergency, Priority or Routine precedence and thus assigned to three separate queue levels. SWITCHED, FUNCTIONAL, and CONTROL messages are queued first-in/first-out within type and by queue. Messages are assigned from the top of each queue to SWITCHED, FUNCTIONAL and CONTROL message processing programs which may operate concurrently.

Subsystem messages are assigned to the remaining queues depending on assigned station priorities and the weight assigned to the type of the Subsystem message. Subsystem messages have one other processing level that is reserved for messages designated as "express" priority by the originating operator. Processing assignments are made starting with the highest priority; with periodic modifications made to the processing assignment scheme such that the lower priority categories are serviced.

On-line modifications of processing levels should be provided to authorized operators. Thus, order of precedence for CONTROL, FUNCTIONAL and SWITCHED messages can be changed and the processing order for subsystem message types may also be regulated.

Recovery

Communications must be supported by message recovery software which is activated in response to an equipment or other failure which significantly interrupts normal message processing. The basic tasks of message recovery software include: notifying active terminals that a message processing failure has occurred and identifying the last complete message received from and transmitted to each terminal; initiating processing of the backlog of complete messages; and, reinitiating transmission of those output messages partially transmitted at the time of the failure.

The message processing environment is usually reconstructed following a failure through use of input/output message logs stored on magnetic tape or random access devices and message processing status records maintained in computer memory or on peripheral devices (checkpoints).

Three basic techniques to provide the reinitiation of system on-line operations are: system empty—no message pending for processing or output (Cold Start); system to be restarted following loss or destruction of main storage information (Checkpoint Restart); and system to be restarted following loss or destruction of peripheral message storage information (Communications Log Restart).

The Cold Start procedure opens all lines and initiates the system to accept input traffic.

The Checkpoint Restart procedure restores the contents of main storage from checkpoint records constructed periodically during on-line operations. The checkpoint records are stored in a mass storage file.

The Communications Log Restart procedure reconstructs and reinitiates input traffic by reloading peripheral message storage from the contents of the communication log tape. The communications log tape contains a copy of each message input to the system.

All active terminals are to be notified when the system resumes on-line operation.

Communications support facilities

To succinctly express the communications support facilities required, the North Carolina Police Information Network specified that "The communications

control must include an integrated system of routines to control all real-time message switching/data communications functions." These routines must be adaptable to the particular hardware environment being proposed and must provide, at least, the following facilities:

- (1) *Line Control*: Communications line control, including sending/receiving bits or characters from/to line control equipment.
- (2) *Message Assembly*: Assembly of messages from all terminals including buffer allocation and queuing of completed input and output messages.
- (3) *Message Queues*: Multiple queues for input must exist with message placement dependent upon at least the sending terminal and processing routine involved. Messages should be retrievable from queues by processing routines via simple logical I/O commands.
- (4) *Polling*: Automatic polling of terminals with the ability to easily modify the polling list and/or the polling sequence. There must be a capability of adding automatic dial-up facilities as needed.
- (5) *Terminal Addressing*: Addressing of individual terminals on shared transmission lines as well as groups of terminals for controlled line usage, employing one address code.
- (6) *Traffic Queuing*: Traffic queuing for inoperative or closed terminals and lines.
- (7) *Automatic Transmission*: Automatic transmission upon terminal/line-up condition.
- (8) *Header Analysis*: Message header analysis routines for determination destination, e.g., another terminal, other terminals or a processing queue.
- (9) *Message Formatting*: Formatting of messages and replies based upon file access results and other system responses.
- (10) *Message Sequence Numbers*: Automatic assignment of sequence/serial numbers to incoming and outgoing messages.
- (11) *Message Validation*: Validation of source and destination codes and message formats. Should an error arise, the system will automatically and according to a prescribed procedure either return the message to the originator, correct the error, or place the message on an intercept station for correction.
- (12) *Code Translation*: Translation between external transmission code and internal processor code if not done via hardware, including editing function of removing non-data characters.
- (13) *Message Logging*: Logging of all messages for on-line retrieval within twenty-four (24) hours of transmission.
- (14) *Error Checking/Recovery*: Transmission error checking and recovery routines.
- (15) *Date/Time Stamping*: Date and time stamping of all messages flowing through the system.
- (16) *Status Reporting*: Line, network, terminal and system status reporting.
- (17) *Automatic Testing*: Automatic procedures to test all terminals for possible malfunctioning and presentation of fault conditions to operator for immediate attention.
- (18) *Computer Interfaces*: High speed communications with other computer systems. Although the computer links are considered for the most part as another type of terminal, significant differences exist in the speed and method of transmission and reception.
- (19) *Message Intercept*: The ability to intercept any message, in a collective manner, which passes through the system will be provided for the system operator. The intercepted message will be displayed for action by a system operator. Modifications in the criteria for interception will be easily implementable in a timely manner.
- (20) *Acknowledgment*: All messages received by the system will be acknowledged. Such acknowledgment will be transmitted to the sender. The acknowledgment format will include the systems and originator's identification numbers and time of receipt.
- (21) *Control Messages*: Accept control messages from command terminal and modify or report operation.
- (22) *Processing Schedules*: Schedule message processing based upon message type, FIFO queue discipline, etc.
- (23) *Interrupt Capability*: Full interrupt accommodation to analyze cause of interrupt by supervisory program and giving control to a specific routine. To take the appropriate action (i.e., issue an I/O when the previous one completes; switching buffers to receive another segment of a message being received, restart polling sequence when a line becomes free).
- (24) *Controlled Time Initiated Actions*: Initiating a given action after a given elapsed period.
- (25) *Message Security*: Ensuring that queued messages are not overwritten incorrectly by testing programs or by errors in operational programs and that messages will not be lost due to various types of machine failure.
- (26) *Fault Indication and Control*: Taking appropriate action when errors of all types are detected, logging and correcting them when possible.
- (27) *Diagnostics and Reliability Checks*: Operating

on-line diagnostics for dealing with errors, for increasing confidence in the system and for assistance to the equipment engineers.

- (28) *Fall-Back*: Organizing a degraded mode of operation when a component of the system fails, for example, a communication buffer. Switching to the redundant or backup component. Organizing recovery from fall-back.
- (29) *System Testing Aids*: Routines to aid in real-time program debugging and test of the communications system.
- (30) *Switch-Over*: Organizing switchover of the combined Information Retrieval and Communication support function to another processor in the event of hardware malfunction.

INQUIRY

Various forms of messages are used for information retrieval and update operations within the Inquiry Subsystems. The requests to the Subsystems fall into the two general system message categories of QUERY messages and UPDATE messages.

Usually Vehicle, Property and Person Application Subsystems are provided for the query and updating of data maintained in the vehicle file, property file and name-want/warrant files respectively.

File oriented request messages are processed within the appropriate Inquiry Subsystem through the use of general task-oriented functions. For each request message, several of the following tasks must be performed: edit and validate the request; verify access authority; formulate file search strategy and record matching profiles; perform the requested search or update; generate output responses (including error diagnostics if appropriate); and, forward the transaction to NCIC as required.

Query messages

The QUERY Request is used to determine whether or not a record exists in the file. If the record exists in the file, the appropriate data is returned to the requesting operator.

QUERY persons should exist in two forms: QUERY PERSONS IDENTIFICATION (QPI), which restricts access to the Master Name File only, and QUERY PERSONS RETRIEVAL (QPR), which goes beyond the Master Name File and automatically retrieves records from subsidiary files such as the Want/Warrant and Criminal History files.

Both QUERY Persons types should permit any mixture of search criteria from among those data

elements carried in the agency specified Master Name File.

General QUERY requests should allow any mixture of search criteria from among those variables carried in the agency specified data record. As large a portion of the file as is possible should be eliminated from the search as determined by the values supplied with the request. The presence of a unique identifier or any of the optional cross-indexed variables will improve this search limiting effect. Both "weighted" profile and absolute identifier matches should be used as appropriate when comparing records for finds.

The search strategies and file organization should be oriented toward efficient operation under conditions where multiple records in the data base will satisfy the search criteria. Thus, considerable effort should be directed toward organizing and structuring the data base and search techniques to minimize the number of accesses required to satisfy the request completely.

Every request message must contain sufficient information to determine a list of one or more file groups which could contain the desired record(s). If not, the full file may be scanned (generally only permitted at agency option). File groups may be found by making a find in a cross-index or by the inclusion of the group defining element(s) in the request message.

In those instances where the agency maintains a cross-index to the file by a particular variable, file search time is significantly reduced if the indexed value is supplied in the request. This function searches cross-indices for each such value present in the message. When indexed values are found, the resulting file group numbers are added to the search group list. Failing to find such values in the index does not necessarily mean that a record cannot be found. In such a case, the normal search strategy is pursued using other variables given in the search profile.

Once the search group list is determined, further definition is needed to select candidate records as each indicated group is scanned. Two methods of matching are usually utilized by the Subsystems. The first uses all unique, positive identifiers from the request and places them in a special argument list. Any record encountered in the search which matches any one of these arguments will be included in the output. Typical of such elements are Social Security Number, Vehicle Identification Number, FBI Number, Drivers License Number, or System Identification Number.

The second technique of record matching involves a weighted element profile. Weights for all possible data elements in the record may be prespecified by the agency. For each QUERY Request, the sum of these weights is computed for all variables indicated in the request. (This combined weight must exceed an agency

specified threshold before a profile type search will be performed.) A match-hit threshold is computed from the combined weight and an agency specified percentage.

Each record scanned will have the values for those variables included in the QUERY Request compared to the values in the record. The weights for each correct variable match are accumulated. The total weight for the record is compared to the match hit threshold. Records with scores greater than the threshold will be selected. The profile type match is made only on those records which were not already selected via special arguments. By varying the element weights, thresholds, and other parameters the agency can provide its own unique balance among the importance of individual elements.

Update messages

The ENTER Request adds a new record to the data file. The ADD, MODIFY, and DELETE Requests selectively alter the content of any number of data elements in a specified record already in the file. The CLEAR (Locate) Request effectively changes the status of a specified record when its subject (person, vehicle, or property) has been located or apprehended. The originator of the record should be automatically notified. The DELETE Request also allows deletion of a specified record from the file.

ENTER request processing

ENTER Requests involve one or more of the following operations. A pre-enter Search is made to avoid duplication. An optional override may be provided to the operator so that the entry of a record can be made even though it is very similar to one already existing in the file. A new record is constructed using element values supplied in the request and element values self-generated by the system. The new record is inserted in its appropriate file group by File Control and all available values are inserted in the appropriate cross-indices. The originator is notified of successful entry combined with a copy of the complete or partial record content. When specified, an NCIC entry will be constructed and sent.

ADD, MODIFY and DELETE request processing

The Subsystem processing performed by these message types consist of the following functions.

Unique and absolute identification of the subject record must be established. The basis of identification

will normally be the record's System Identification Number plus any other required variables established by the agency as being mandatory. Optional agency-specified rules are then applied for authorization to up-date the record (such as requiring that the requesting terminal be the same as the record originator). The requested operation is performed on the record for each element specified in the request and element-by-element editing of the requested type of operation is allowed for each.

File control is used to replace the updated record in its appropriate file group and file cross-indices entries are removed, added or modified if such elements are involved. The originator is notified of successful completion along with a copy of the updated record. NCIC transactions will be constructed and sent when required.

The capability to delete an entire record is also provided. This function differs from the ordinary DELETE request described above with ADD and MODIFY in that more stringent criteria can be applied to determine the authorization. Furthermore, the entire record may be immediately deleted via File Control or optionally, the record may be flagged for removal during the next periodic purge run (Background). All such flagged records would be omitted from normal on-line responses until they are removed.

CLEAR (Locate) request processing

The Subsystems for this type of request modify the status of the record to show that it is inactive and information is added to the record to indicate the time and source of the "locate" action and event. According to conditions and options established by the agency a notification message is constructed and sent to the originator of the record indicating the subject record and the nature of the "locate" event. The request originator is advised of the successful update and notification along with a copy of the updated record. When so specified, NCIC is advised of the "locate" event.

NCIC translate function

Each Subsystem request-type should contain specific logic to determine whether an NCIC message can and should be formulated. Should NCIC be referenced, a general NCIC translate function should be provided to select elements from among those specified in the request, translate their values to NCIC codes, and place them in acceptable NCIC format. Subsystem and NCIC responses are independently sent to the origi-

nator. NCIC responses, which may be delayed, are usually directly routed to the originator without Subsystem intervention. With this translate function, NCIC request can be formulated by the user in a non-NCIC specific format and reformatted by the computer for NCIC acceptance. This eliminates the use of two separate formats or the required adoption of the NCIC format by the system.

FILE MANAGEMENT

Law Enforcement File Control supports the Inquiry Subsystems by providing the logical record storage and retrieval operations which permit the Subsystems to be independent of the physical characteristics of the files being processed. It provides access control to preserve the security and integrity of System data, and also supervises and schedules all requests for file access from Subsystems. In so doing, it is responsible for preventing simultaneous file access requests from Subsystems by applying appropriate priority recognition and delay criteria. File Control also provides the mechanisms required for system file recovery.

When a single generalized concept for the organization, and access to all on-line files is utilized, all on-line files are structured similarly. Each contains an agency specified set of Data Elements arranged in a format to produce a record type. Multiple variable-length records are in turn "blocked" according to a common characteristic. Multiple blocks may be linked together when necessary to contain an entire sub-group of the file.

To provide rapid and efficient access to specific records within the file, multiple cross-indices to individual files may be defined and maintained. A cross-index must be based on some unique identifier of the subject vehicle, person or property (such as Vehicle Identification Number (VIN) or Social Security Number).

Organization of a file consists of using one or more data elements (variables) within the record to partition the file into a number of groups such that in a preponderance of access situations few groups need to be scanned to find a specific record or set of candidate records. Furthermore, the group's size may be set such that the group may easily be scanned within the agency's desired limits of maximum response time for an individual file and type of request. Thus, full file scans are minimized and may, at user agency discretion, be disallowed.

The usual variables selected for defining on-line file groups are:

Vehicle File—License Plate Numbers & State of Registration

Master Name File—Phonetic Name Codes

Want/Warrant File—Corresponds to Master Name File

Property File—Property Category and Serial Number

File Control should provide all Subsystems with a logical file processing capability allowing the following functions to be performed:

OPEN File—initializes processing within the designated file for the requesting Subsystem.

CLOSE FILE—terminates processing of the designated file for the requesting Subsystem.

READ Record—the first such request causes the first record of the file, or designated record group, to be transferred from the input buffer into the call Subsystem's designated core storage area. Each subsequent read causes the next logical record to be transferred to the designated storage area.

LOCATE READ Record—the LOCATE READ function operates in the same manner as READ described above with the exception that no movement of the record within core storage from I/O buffer to Subsystem workspace will take place. Instead, the Subsystem is provided with the location of the record. The use of LOCATE READ enables more rapid scanning of records by the Subsystem.

REREAD Record—causes File Control to transfer to the calling Subsystem's core storage space the same record as that just read.

DELETE Record—causes File Control to delete the last record read from the file.

REPLACE Record—causes File Control to replace the record last read with the record designated in the request.

INSERT Record—causes File Control to insert a designated record in the file.

In addition to the service capabilities described above, File Control should also provide the following features: access validation, multiprogramming capability, data file integrity, file recovery, test mode and overall storage management.

CONCLUSION

Real-Time Law Enforcement and Criminal Justice information systems are almost becoming commonplace as more and more agencies acquire computerized communication and inquiry systems. There is evolving a network of computers and communication systems that will tie all of the law enforcement system together.

The current National Crime Information Center (NCIC) of the FBI makes available to each state and major urban agency data on wanted persons, stolen vehicles and stolen articles. Each of the real-time law enforcement systems are interfaced with this central FBI computer system, giving all of the terminals in each system direct access to the NCIC files. A nationwide criminal history system is under development that will connect each of the participating states via computer communications to a master name index and criminal profiles on all major offenders in the United States.

The era of large, complex and integrated communication and inquiry systems has been projected as a mid-1970's phenomenon. With the current emphasis and acquisitions in the Law Enforcement field, it looks like they will again lead the industry in Real-Time computerized communication and inquiry systems as they did in land-mobile radio and teletype torn-tape message switching systems.

REFERENCES

- 1 *Colorado's automated law enforcement system request for proposal*
Department of Administration No 10 July 1 1970
Denver Colorado
- 2 *Communication and inquiry general description, law enforcement application program*
UNIVAC Division of the Sperry Rand Corporation
February 1971 Blue Bell Pennsylvania
- 3 *Electronic message switching system specification for Texas Department of Public Safety*
No DPS-8-2-4 April 23 1970 Austin Texas
- 4 *Florida crime information center request for proposal*
May 1 1969 Tallahassee Florida
- 5 *Metropolitan Government of Nashville and Davidson County Police Department request for bid for a police information network system*
February 19 1971 Nashville Tennessee
- 6 *Missouri law enforcement data system specifications*
Missouri State Patrol April 1968 Jefferson City Missouri
- 7 *New Jersey statewide communications and computer base information system for the Department of Law and public safety request for proposal*
No 5-71 January 29 1971 West Trenton New Jersey
- 8 *New York State police real-time law enforcement information system specifications*
January 1970 Albany New York
- 9 *North Carolina police information network request for bid*
June 1970 Raleigh North Carolina
- 10 *Ontario police tactical information centre electronic data processing system specifications*
December 1968 Ontario Provincial Police Toronto Canada
- 11 *Oregon law enforcement data system specifications*
December 12 1969 Salem Oregon
- 12 C T SMITH
A computerized national law enforcement communications system
Law Enforcement Science and Technology Volume III
ITT Research Institute 1970
- 13 *Washington State patrol computer message switching system request for proposal*
July 31 1970 Olympia Washington
- 14 P M WHISENAND J D HODGES JR
Automated police information systems—A survey
Datamation May 1969

The Long Beach public safety information subsystem

by GEORGE M. MEDAK

City of Long Beach
Long Beach, California

and

DR. PAUL M. WHISENAND

Institute for Police Studies
Long Beach, California

and

GARY GACK

Digital Resources Corporation
Charleston, West Virginia

INTRODUCTION

Not many years ago, officials in a city with a five year capital improvement plan, a "master" plan, and a computer that handled payroll calculations and utility billing were termed relatively progressive. This was considered a dynamic approach to the use of modern management techniques and computer technology.

In more recent years there has been a dramatic increase in the challenge of city management. A growing crime rate, financial dilemmas, problems with ecology, urban transportation and the like provide great tests of management skill for today's municipal executive. Information about people and the urban environment is a critical requirement for effective municipal management.

It is recognized that the development of a comprehensive municipal information system is a costly and complex task. An independent effort by a municipality to research, develop, and implement a system would require substantial technological resources, together with a large appropriation of local tax dollars to fund the project. Furthermore, individual projects across the nation would lead to great duplication of effort, use of resources and cost.

It was with these facts in mind that the major federal agencies concerned with urban programs established a coordinated research effort to develop municipal information systems. The Urban Information Systems Inter-Agency Committee (USAC) Program was estab-

lished on September 10, 1968, by the Secretary of the Department of Housing and Urban Development. Sponsoring the program were representatives from nine federal agencies:

- Department of Housing and Urban Development
- Department of Justice
- Department of Transportation
- Department of Labor
- Department of Commerce
- Department of Health, Education and Welfare
- Bureau of the Budget
- Office of Economic Opportunity
- Department of the Army, Office of Civil Defense

The USAC Program, supported by multi-agency funding, initiated two classes of effort. One class, a 3 year program directed at total integrated municipal information systems, and a second 2 year effort directed toward functional subsystems. The functional subsystems as defined by USAC were: (1) Public Safety; (2) Physical and Economic Development; (3) Public Finance; and (4) Human Resources Development.

During July, 1969, the Department of Housing and Urban Development, acting on behalf of USAC, initiated a nation-wide procurement action inviting 250 cities between the population of 50,000 and 500,000 to compete for the two classes of contracts. Approximately 100 proposals were received from 79 cities in 30 states. On January 13, 1970, six awards were announced by

TABLE I—USAC Municipal Information System Projects

System/Subsystem	City
Total Integrated Municipal Information System	Wichita Falls, Texas and Charlotte, North Carolina
Public Safety	Long Beach, California
Physical and Economic Development	Reading, Pennsylvania
Public Finance	Dayton, Ohio
Human Resources Development	St. Paul, Minnesota

the Department of Housing and Urban Development for the projects and cities shown in Table I.

A unique requirement in the request for proposals was that the government called for the creation of a consortium by all respondents. The consortium had to consist of a municipal government (as the prime contractor), a systems/software firm (as a subcontractor), and a university/research center (as a subcontractor).

The City of Long Beach, California, awarded the prime contract by HUD for development of the Public Safety Subsystem, has the primary responsibility to the federal government. Long Beach must organize, monitor and supervise the overall project to assure that all contractual performance requirements are fulfilled. The City has also made a substantial in-kind contribution of personnel and other resources.

The Project Director is an employee of the City and works closely with a Management Steering Committee established specifically for the project. In addition, the City has advisory and technical personnel assigned to the effort to assure that system development tasks lead to operational features which will best satisfy the needs of the various city user organizations.

Digital Resources Corporation (DRC) is the Systems Contractor and is responsible for the planning and execution of technical efforts associated with the project. Specific areas of responsibility include: analysis of the current operations; design of the data base structures; and design, development and implementation of the computer applications. The contractor will inventory and analyze the current technology in information management systems. That technology will be employed to conceptualize and design the system for the City of Long Beach.

The Institute for Police Studies (IPS), California State College at Long Beach, is the third member of the consortium. Through the Institute, a team of public safety authorities is providing consulting services to the project. IPS is responsible for technical leadership

related to the orientation and training process. The Institute will also perform project monitoring and evaluation to determine overall effectiveness of the developed system.

PROJECT ORIENTATION

The USAC Public Safety Subsystem project under way in Long Beach was initiated in March, 1970, and covers the complete development cycle in a 24 month period. It began with system analysis and includes system conceptualization, design, development, implementation, and evaluation. The project is chartered so as to result in a significant state-of-the-art advance in municipal information systems. It is not envisioned as a pure research project, however, but is committed to reaching operational status.

To maximize the utility of this project to other municipalities, all stages of conceptualization, design, development and implementation are being conducted with due consideration for system characteristics that enhance transferability.

To facilitate dissemination of information on activities relative to experience gained in the six project cities, HUD requested that all projects be organized into specific tasks. These tasks, as defined by the federal government are:

- Analysis—The project team will perform an in-depth analysis of the present operation of the Police, Fire, Civil Defense and Licensing/Code Enforcement agencies within the City of Long Beach.
- Conceptualization—Conceptualize the emerging information system including both manual and automated components.
- Design—Design in detail those system components selected for implementation by the City.
- Development—Complete programming and debugging tasks as required to achieve operational status.
- Implementation—Successfully maintain operation of the developed applications in the operational environment.
- Orientation and Training—Orient and train the affected personnel in the effective use of the system.

During the initial 12 months of activity, efforts were applied principally to the tasks of project organization, system analysis and system conceptualization. Excellent progress has been achieved to date. The following paragraphs provide a brief report on these activities.

ANALYSIS OVERVIEW

The objectives of systems analysis as stated in the Long Beach contract with HUD are:

- Analysis of the Municipal Governmental System—Broad examination of municipal affairs and the relationship of the Public Safety functions to those operations and other governmental agencies.
- Analysis of Current Operations—Detailed review of current information processes with a view toward attaining an in-depth knowledge of information carriers, flows, procedures, policies and requirements.
- Analysis of Decisions—Identification of decisions and the inter-relationship of those decisions currently made in the Public Safety functions.
- Analysis of Information Requirements—Definition of the existing data elements and their relationship to the decision making process.

The System Analysis Task findings, which were submitted to HUD in a 5-volume, 3700 page report, provide the foundation for subsequent project efforts. The significance of this effort is in the establishment of an accurate data base reflecting departmental needs. This data base is an essential information resource in designing an improved information system that is responsive to the operational needs of the city. Through a thorough analysis, we are assured that the Public Safety Information Subsystem now under development is geared to actual information needs, rather than hypothetical requirements.

Computer aided analysis techniques

Analysis of the information flow through the agency on both an interdepartment and intradepartment level is a burdensome and time consuming job. As a result, the public safety project staff developed a complete system of computer programs called META DATA/1 to aid in the analysis of both content and flow of information in the public safety function. Capable of operation on current third generation computers, the META DATA/1 system consists of two phases, each of which begins with a systematic survey or interview procedure and culminates with a series of reports and a tape file of descriptive information about the data elements.

Phase I is primarily concerned with the forms and elements of data currently used in the agency. These forms (sets) and their composing elements are processed into a series of reports which are used in systems

analysis and subsequent tasks. Phase II is concerned with the flow of sets through the agency and costs (total and by functional activity) of processing the respective sets. The resultant report provides subtotals at various levels and grand totals at the end of report are printed for the following quantities: (1) average monthly volume, (2) accumulated quantity, and (3) average monthly cost.

The elapsed time required and the cost of processing information at each functional operation is clearly identified. The application of the META DATA/1 to other localities represents another transferable feature in the planning, design, and implementation of municipal information systems and functional systems for public safety.

One of the outputs produced by META DATA/1, Phase I, used in the Police Department, is a listing in data element name sequence of each element on each of the 1088 forms identified. This automated processing of forms and data elements enabled the staff to uncover some very interesting facts relative to the Police Function in Long Beach including the following:

- There are more forms (1088) than employees (877).
- There are a total of 12,944 data elements in all forms.
- There are 384 data elements termed "NAME," with 10 qualifiers (arrestee, employee, applicant, suspect, etc.).
- Many synonyms are used to describe the same data element (hand gun, pistol, automatic, revolver).

The Phase II outputs, which include a record of the handling of each form, are even more revealing. These reports provided for the following observations:

- The Police Department use 1048 separate files of all types.
- It costs approximately \$170,000 per month, or 19 percent of the budget to process all forms.
- Five forms account for 35 percent of the cost (Crime Report, Field Report, Accident Report, Parking Citation, Arrest Report).
- Ten forms (1 percent of the total) account for 50 percent of the cost.

Perhaps even more significant than these isolated facts is the design tool which was available as a result of the automation of this data. First, these reports provide an objective indication as to where the highest potential benefit of automation lies. Second, they provide a complete cross reference of every file and every

organizational unit which is concerned with any given form. This makes it possible to examine the entire effect of any change in form design or process flow. Third, the META DATA/1 outputs also provide a complete cross reference of every form and every file which is used by any given organizational unit. This makes it possible to examine the entire impact of any organizational change on the information system.

CONCEPTUALIZATION OVERVIEW

The project efforts are currently directed toward detailed design of the applications which will be implemented to demonstrate the USAC philosophy and provide the City of Long Beach with certain operational capabilities. The concepts upon which applications are being designed are derived from the Conceptualization Task Completion Report recently submitted to HUD.

The conceptualized Public Safety Information Subsystem (PSIS) for the City of Long Beach adheres to the general guidelines described in the HUD Request for Proposals. To obtain a better understanding of what municipal functions are included in the Public Safety System, a brief discussion of the total municipal information processes will be given followed by a definition of those elements contained in the conceptualization.

As defined by USAC, a total integrated municipal information system can be viewed as consisting of four functionally oriented subsystems. When viewed collectively, the subsystems function in a united manner and show a high degree of horizontal information interchange. The concept of municipal information subsystems was introduced in order to separate the total system into manageable, functional groupings which could be incrementally developed.

To further define the makeup of an information subsystem, each is comprised of a group of related functions. Functions within the context of this definition may or may not follow organizational lines. They are, however, characterized by the processing of information to accomplish a specific goal, i.e., assessing, planning, etc. In the Public Safety Subsystem, the functions are identified as Police, Fire, Civil Defense and those aspects of Licensing and Code Enforcement which are applicable to the other three functions.

For further refinement, each function is thought of as being composed on one or more components. The component defines the points where similar information is input, processed and output from the subsystems. Primary emphasis was placed throughout the Conceptualization Task on the development of the concepts

TABLE II—Public Safety Subsystem Functions and Components

Function	Component
Police	Case Reporting
	In-Custody
	Traffic Reporting
	Investigation Support
	Calls-for-Service
	Wants/Warrants
Fire	Vehicle/License
	Stolen Property
	Fire Suppression
	Fire Investigation
License/Inspection/Code Enforcement	Fire Prevention
	Fire Dispatching
	Fire Code Enforcement
Civil Defense	Police Permit and Licensing
	Civil Defense Shelter Licensing
	Shelter Management
	Resources

for components since they represent the operational elements of the information subsystems. Table II is a listing of the functions and their related components as identified in Long Beach.

Each of the components is defined as a logical implementation block which can be developed and implemented either singularly or in conjunction with several other components.

Public safety horizontal and vertical interfaces: design concepts

The vertical subsystem may be described as the subsystem which links various levels of government along functional lines. Within public safety, the vertical relationships between city, county, region, state, and federal levels of government are well defined and structured as compared to other similar municipal relationships.

The most pronounced vertical intergovernmental relationship is in the Police Information function because of presently established interfaces between the regional, statewide, and federal criminal justice information systems. For example, the Long Beach system must interface as shown in Table III.

Public Safety is a subsystem to not one, but many information systems at various levels of government. It does become a logical building block for the other systems since it is operational in the municipal government structure.

USAC has stated that "as a matter of emphasis, this project is aimed at the discovery, establishment and automation of the horizontal subsystem."

A horizontal subsystem may be described as a set of data linkages which exists between one subsystem and another, one function and another, or one component and another. These linkages appear in three primary forms:

- Informal—phone calls, meetings, etc.
- Formal hardcopy interchange—memos, reports, etc.
- Data Sharing

The latter form is the one most subject to automation and is, therefore, the prime target of project effort. A three step approach to the development of design concepts was adopted.

Subsystem concept

In order to perceive the rather complex array of interfaces to which the Public Safety Subsystem contributes, it is necessary to examine each function of each subsystem at the component level. For ease of understanding, the project team has chosen to classify components as to their nature. Components are identified according to the following classes:

- Supportive components—Those information processes concerned with the internal operations of one or more functions.

TABLE III—Typical Systems Interface

Government Level	System
Federal	NCIC (National Crime Information Center) SEARCH (System for the Electronic Analysis and Retrieval of Criminal Histories)
State	Auto-Statix (California Highway Patrol) DMV-AMIS (Department of Motor Vehicles—Automated Management Information System) CJIS (California Criminal Justice Information System)
Regional	RJIS (Los Angeles Regional Justice Information System) AWWS (Los Angeles Automated Want/Warrant System)

TABLE IV—Horizontal Linkages

Public Safety Horizontal Interfaces	Supportive	Environmental	Operational
Human Resources		People Data Base	
Physical and Economic Development		Property Data Base	
Finance	Fiscal Data Base		
Public Safety			People Data Base Property Data Base

- Environmental components—Those information processes concerned with collection and maintenance of data which describes the community or environment to be served.
- Operational components—Those information processes which are triggered by specific events such as crimes, fires and so forth.

In view of the fact that those components defined as supportive and environmental in relation to public safety are also operational components in other subsystems, they were not included in the conceptualization of the Public Safety Subsystem. Table IV illustrates the horizontal linkages which exist in relation to public safety at the subsystem level.

As indicated in Table IV, the various data bases are the mechanism which implements data sharing. In other words, data base concepts are employed because they are in fact an operational necessity of the system. The technical benefits derived are only incidental and were not the motivating factor behind this approach.

Function/component concept

After arriving at a general conception of horizontal relationships at the subsystem level, the project team looked next at the function level and at components within each function. The perspective of the data bases as a data sharing mechanism was maintained. The rationale behind the identification of components as presented is too lengthy to go into here, but the result of this step is illustrated in Figure 1.

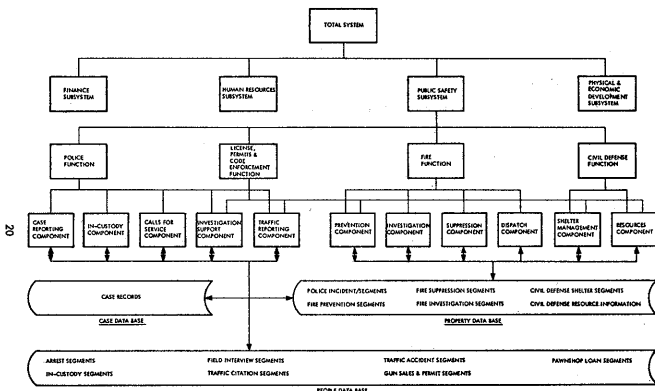


Figure 1—Public safety subsystem conceptualization

Data base concept

To implement the subsystem function and component design, we have reached a point at which information technology must be employed to the full extent of its capabilities. Without attempting to address the *organizational* issue of centralization, it does seem clear that *informational* centralization is necessary to implement the Long Beach Public Safety concept. Informational centralization, in contemporary terms, is an integrated data base.

The Data Base Management Software (DBMS) technology to be employed in Long Beach has three primary attributes which are essential to informational centralization.

The first attribute, of which much has been written and little has been done, is the "integrated data base." The integrated data base technique of storing data in a computer system differs from the traditional approach

in terms of the means used to associate elements of data together. It has been traditional to associate data together in terms of use. In contrast, the integrated data base approach suggests that data be associated by object.

There is considerable evidence that association based on object will reduce operational costs associated with redundant acquisition, storage and maintenance of data while at the same time improve the currency, accuracy and reliability of the data. The second attribute provides the capability for the information system to tolerate change.

Development of an integrated information system, of the scope anticipated for the Long Beach Public Safety Subsystem, requires an incremental approach. This approach necessarily assumes that additions may be made without major modifications to initial programs or components. Were this not the case, an incremental development would be economically impractical.

To resolve the problem, data definitions are separated from the program. This approach creates a software environment which supports a single data definition, known as a "data dictionary", that is shared by all programs accessing the file.

The data dictionary facilitates the development of a system which permits independent evolution of both the data files and programs without excessive modifications to either, as only the dictionary requires maintenance, not the programs.

In keeping with the idea of organizing data by object, the Public Safety Subsystem employs two primary data bases: People and Property.

The Property data base (Figure 2) acts as a data sharing mechanism at the subsystem level between all four subsystems, and the function level within Public

DATA BASE: PROPERTY		Physiol & Economic Development		Finance		Human Resource Development			
SUBSYSTEM: PUBLIC SAFETY									
Segment:	Common Identifiers	Fire Suppression	Fire Prevention	Fire Investigation	C.D. Shelter Management	Police Calls for Service	Utility Billing, etc.	Tax Accounting, etc.	Health Etc.
Data Elements:	Address Corner, Fire Zone, Police Reporting Area (Start) Etc.	Date, Incident No, Material Ignited, Fire Company, Responding Etc.	Permits, Date of Invtly, Complete, Property Class, Violation Type, Correction Type Etc.	Date, District No., Area of Fire Origin, Building Occupant, Property Value, C.D. Shelter, Source & Form of Ignition Etc.	Standard Loc., Shelter, Shelter Spaces by J.P., Company, Owner's address, Facility No., Shelter, Sheds Etc.	Incident Type, Date, Time Etc.	Owner, Occupant, Service Etc.	Value, Time Etc.	
Components:	Fire Dispatch	Use	Use	Use					
	Fire Suppression	Update Use							
	Fire Prevention		Update Use						
	Fire Investigation			Update Use					
	C.D. Shelter Management	Use	Use		Update Use				
	C.D. Services		Use						
	Police Calls for Service								

Figure 2—Segment detail of a property data base

DATA BASE: PEOPLE		Physiol & Economic Development		Finance		Human Resource Development					
SUBSYSTEM: PUBLIC SAFETY											
Segment:	Common Identifiers	Arrest	In-Custody	F.I.	Moving violation citation	Accident Report	Prom Shop Loan	Own Sale	Health Record	Utility Billing, etc.	Reynold/Personnel/ Etc.
Data Elements:	Name, Age, Sex, Address Etc.	Date, Charge, DR No., Etc.	Booking No., Bail, Evidence, Etc.	Physical Description, Date, Time, Location, Etc.	Charge, Date, Time, Location, Etc.	Date, Amount, Date, Etc.	Item, Amount, Date, Etc.	Serial No., Caliber, Type, Etc.	Districts, Party, Etc.	Qty., Amount, Etc.	Evidence, FICA, With Holding, Etc.
Components:	Police In-Custody	Update Use	Update Use								
	Police Case Reporting	Use	Use			Use					
	Police Investigation Support	Use		Update Use	Use	Use	Use	Use			
	Police Calls for Service	Use	Use	Use	Use	Use	Use	Use			
	Police Traffic Reporting					Update Use					

Figure 3—Segment detail of a people data base

Safety between Fire, Police and Civil Defense and at the component level within the Fire Function between Dispatch, Suppression, Prevention and Investigation.

The People data base (Figure 3) acts as a data sharing mechanism at the subsystem level between all four subsystems, and at the component level within the Police Function between in-Custody, Case Reporting, Investigation Support, Calls for Service and Traffic Reporting.

PERSPECTIVE

During the remainder of this project, Long Beach will proceed with the phased implementation of selected

applications to substantiate the hypothesis proposed by the concepts developed. The conceptualization described above is expected to evolve into a multi-year implementation plan for the City of Long Beach and become the basis for planning in other municipalities.

If the Long Beach Public Safety Subsystem is successfully transferred to another jurisdiction, the USAC objective of transferability will be validated. This can occur only if the recipient municipality openly approaches change to its existing policies and operations. Such change appears to be desirable in view of anticipated benefits to be derived from the implementation of an integrated municipal information system or subsystem.

State criminal justice information systems

by ROBERT R. J. GALLATI

New York State Identification and Intelligence System
Albany, New York

INTRODUCTION

There has finally been wide recognition of the need to improve the systemic relationships of the various functions and processes of what has been euphemistically referred to as our criminal justice system. With this recognition has come an understanding of the central role of criminal identification bureaus in computerized criminal justice information systems, which, in turn, serve as foundations for the development of true criminal justice systems.

As a practical matter the state is the most logical governmental level at which computerized criminal identification bureaus could be housed. Local communities, regardless of size, necessarily have less complete files than those at the state level. Criminal law, both in terms of enactment and enforcement, is state-level based. State identification files are records of violations of the criminal laws of the particular state involved. As an operational matter, it is exceedingly difficult for a national agency to handle the fantastic workload involved in an attempt to process fingerprints and perform other identification functions for the entire nation.

An additional factor which commends the maintenance of computerized criminal justice information systems and the identification bureaus they are structured around at the state level is the increasing public concern about possible invasions of privacy involved in computer data banks, particularly those at the federal level which might be interfaced with others to form a single giant National Data Bank. Criminal justice information system data banks necessarily contain derogatory records, so it has been strongly recommended by civil libertarians and many criminologists that comprehensive criminal justice information files be kept at state level.

Obviously, if we are to retain our computerized

criminal justice information systems at the state level, there must be some method for the interstate exchange of criminal history records. This was fully recognized during the development of Project SEARCH (System for the Electronic Analysis and Retrieval of Criminal Histories). The FBI/NCIC (National Crime Information Center) has assumed responsibility for maintaining the central index of the SEARCH-type system which becomes operational this November.

The critical role to be played by state identification bureaus in the future of the NCIC Criminal History Record Exchange System is apparent from a policy statement approved on March 31, 1971, at a meeting of the National Crime Information Center (NCIC) Policy Board. The Board determined that in order for the NCIC system to evolve into a truly national system . . . "each state must create a fully operational computerized state criminal history capability within the state. . . ."

It is submitted that state criminal justice information systems and the identification bureaus which form the nuclei of their files, are destined to play an ever-increasing role in the area of public systems dedicated to law enforcement and criminal justice. One of the noteworthy examples of the development of such systems is the New York State Identification and Intelligence System (NYSIIS).

I propose to present the NYSIIS story as an analytic case study of a particular model for a state criminal justice information system. NYSIIS did not evolve from some other agency or function. It was created "de novo" as a conscious effort to produce both an agency and a function that had never before existed in New York State, or elsewhere. It is unique, and may well continue to be the only one of its kind in the nation. However, all 50 states will follow the model in one way or another, even though each may develop an indigenous system, which, on the surface, appears to differ considerably.

NYSIIS CASE STUDY

Origins

NYSIIS was created as an agency in 1965. The concept of NYSIIS rests upon the following basic principles of the unitary nature of criminal justice: all criminal justice agencies need to participate in and share a joint data bank; the submission of information thereto should be primarily voluntary; NYSIIS is to be a service agency only, with no powers, duties or facilities to arrest, prosecute, confine or supervise; security and privacy considerations must permeate the system and involve central and remote NYSIIS operations; new dimensions of science and computer technology can be applied to provide greater effectiveness in filing methodology and the utility of processed data; and that criminological research will be supported by a vast resource of computerized empirical data available for variable searching to test theses, hypotheses, theories and pilot projects, thereby enabling criminal justice administration to evaluate its own procedures, practices and operations.

Development

It was very evident from the start that if NYSIIS was to function effectively as a criminal justice information system serving all functional areas of criminal justice, there were some obvious conditions that had to be met:

1. NYSIIS had to be created and maintained as an independent agency so that it could serve all functions without fear or favor. This has been a public administration and computer sciences problem (opportunity);
2. NYSIIS had to have a vast computer capability and engage in massive historical and ongoing data conversion of the millions of criminal history records contained in its manual identification files. This has been a systems and computer sciences problem (opportunity);
3. NYSIIS had to advance the state-of-the-art of computer-related techniques for the further automation of the fingerprint identification process. This has been a research and development and computer science problem (opportunity);
4. NYSIIS had to create state-of-the-art computer-related technology for the development of new and improved analytical techniques for the identification and intelligence functions. This has been a planning, research and computer sciences problem (opportunity);

5. NYSIIS had to provide computer-related communications systems for remote access to the system data bank and for computer interface with interstate information exchange systems. This has been a systems, communication and computer sciences problem (opportunity);
6. NYSIIS had to be able to survive a period of several years during which it produced only a minimum tangible product in the service of the criminal justice community. This has been a public relations and computer sciences problem (opportunity);
7. NYSIIS had to embrace a sophisticated security and privacy program in order to allay the fears of those who perceived computerized data banks of derogatory data about individuals as a threat to civil liberty. This has been a political and computer-sciences problem (opportunity).

Computer opportunities

It is fair to say that NYSIIS would be as nothing but for its computer capability. In every phase and at every stage of its origins, development and continued growth computer science problems (or opportunities) presented themselves and then became the most constantly viable elements of continued survival. Indeed, the future of NYSIIS and criminal justice information systems is wholly dependent upon computer capabilities and related technology. Perhaps it is more accurate, therefore, to refer to the role of the computer as computer-science opportunities rather than computer-science problems.

Agency independence

A cardinal tenet of the founders of NYSIIS was that it should be an independent agency with its own dedicated computer system. Maintaining bureaucratic independence has been a torturous trail to blaze. As a small agency among the giants (State Police and Department of Correctional Services), it is not surprising that influential legislators would each year call for NYSIIS' elimination and the consolidation of its services, either with the State Police or the Department of Correctional Services. In fact, within a single week, a prominent legislator recommended that NYSIIS be taken over by the State Police on one occasion, and then recommended that it be absorbed by the Department of Correctional Services on another occasion, just a few days later.

Another facet of the war for independence has been misguided attempts to consolidate state agency com-

puters on a statewide basis. To date these have been frustrated, largely because of the fact that NYSIIS seized the opportunity to obtain its own very large computer system as soon as possible. Had NYSIIS yielded to the temptation to "get started" with some attractive police modules such as stolen motor vehicles and stolen property, it may have found itself a ready candidate for absorption by a larger agency or have been required to share a general service computer with a number of other state agencies. Since NYSIIS, from the beginning, went for the "big apple"—a vast computerized criminal history file—very early in the game, it became not so readily digestible and it managed to stand alone and independent—saved by the Burroughs 6500!

Data conversion

Data conversion for the computer is invariably considered a simple problem by people who have never experienced its impact. Those who are veterans of conversion battles know better. They also know it is particularly difficult to convert a very large manual file while that file is being used on a day-to-day basis in essential operations.

In the NYSIIS conversion of criminal history records as is true to a greater or lesser extent in all criminal identification bureau conversions, there have been certain added dimensions that further increased the difficulty attached to such an operation, such as:

1. Different time periods—due to the type of documents involved (fingerprint cards, court dispositions, institution cards, etc.), and primarily for control of data conversion it was necessary to set four time periods from 1927, to the present;
2. Multiplicity of agencies—there are more than 1000 relatively autonomous agencies which have submitted source documents to NYSIIS;
3. Document differences—documents received from varying sources differ in format;
4. Information location—the positioning of information on submitted forms varied from the same and different sources.

At present more than 750,000 criminal history records are on tape and fully edited and purified records are being added to the computer data base at the rate of over 10,000 per month. Obviously, this amount of storage of extensive criminal histories requires tremendous computer capacity and maximum speed and multiprocessing capability. Computer science has provided NYSIIS with these capacities and capabilities

and the opportunity to provide a two-hour response time for fingerprint submissions—as opposed to the 10-14 day response time of the old manual system. This 12,000 percent improvement in response time is completely dependent upon the extensive computerization of the identification function at NYSIIS.

Automated identification

The fingerprint identification process which is the basic function of all criminal identification bureaus virtually demands computerization and complete automation by its very nature. It is fortunate that NYSIIS planners recognized this from the very beginning of the agency. Today, NYSIIS is the most fully automated identification bureau in the world.

The need for systematic improvement in the operation of these bureaus is accentuated by three recent developments:

1. New legal procedures such as preventive detention, release on own recognizance, forthwith sentencing, and mandatory rapid arraignment and bail setting require swift criminal history record responses;
2. The overwhelming increases in the *volume* of fingerprint submissions has resulted in larger and less readily accessed files;
3. Facsimile transmission and other telecommunications devices have eliminated time delays connected with delivery and focused attention upon the lag-time at the point of processing.

The fundamental steps in the fingerprint identification process are as follows:

1. Name search of main file and wanted file
2. Classification of fingerprints
3. Fingerprint search
4. Fingerprint comparison
5. Criminal history preparation

NYSIIS has computerized the name search process, both for wanteds and the main file. To date, no agency in the world has succeeded in automating the classification of fingerprints; however, extensive research has been conducted by the FBI, NYSIIS and others to achieve this objective. NYSIIS, however, *has* developed a computerized fingerprint search technique with the remarkable capability of searching incoming fingerprints against a base file of 2.5 million fingerprint classifications in less than 30 seconds. Fingerprint comparison through microfilm image retrieval techniques is

within the state-of-the-art and NYSIIS intends to obtain such a capability as soon as funds become available. Finally, the hard copy criminal history record is retrieved from the computer data bank and printed out in NYSIIS, or at a remote access facility. Here again, we see that the opportunity to utilize computer sciences and related technology is the key to entirely new dimensions of service which serve to protect civil liberties and to deal more effectively with suspects and apprehended criminals.

Analytical identification

The basic function of an identification bureau (which is, in turn, at the heart of any criminal justice information system) is to receive hard copy sets of fingerprints containing the friction ridges of all ten fingers of the subject; compare these sets with those in the base file and produce verified criminal history records (or "no record responses", as the case may be). However, there are many analytic (investigative) needs of criminal justice agencies which can be satisfied as by-products of the computerized criminal identification bureau. These bonus-type modules of the system have a high pay-off in terms of public support and increased credibility in the criminal justice community. Examples of some of these computerized modules which NYSIIS has developed to date are as follows:

1. Latent fingerprint identification
2. Fraudulent check
3. Personal appearance
4. Warrant/Wanted
5. Organized Crime Intelligence
6. Stolen motor vehicles (Automatic License Plate Scanning)
7. Modus Operandi
8. Criminalistic data analysis

All of these analytical modules are viable theoretically for investigative purposes by all branches of the criminal justice process. However, as a practical matter, they are most often within the police domain and their availability pleases the law enforcement segment of the total spectrum of criminal justice administration. Since 70 percent of the total resources of criminal justice are concentrated in the police function, special attention to law enforcement requirements was definitely in order for this fledgling agency.

However, it was not sufficient merely to take the very primitive files that currently existed and computerize them. This would have outrageously sub-

optimized the capabilities of the computer and the developing criminal justice system. Ergo, NYSIIS found itself in the position of having to create state-of-the-art computer-related technology in order to justify its efforts to meet these law enforcement needs. Once again, great opportunities were presented to provide orders of magnitude improvement in this vital area of government.

For example, there has never before been an effective latent (crime scene) fingerprint identification system. Under the conditions of existing fingerprint classification systems, it is not possible to search the fingerprints of unknown suspects left at the scene of a crime through the millions of sets of prints in the main file. As a result, special files of recidivists in those types of crimes where there are likely to be prints left at the scene and the perpetrators are not otherwise identifiable (i.e., burglary, auto theft, etc.) have been created. The largest file of this type in the United States contains the prints of less than 30,000 persons—compared with many millions in the base file! NYSIIS' studies indicate that at least five percent of all burglaries could be solved through latent print identification if a sufficient number of crime scene fingerprints were lifted and processed in a large base file. (It must be recognized that less than 20 percent of current burglaries are cleared by all other investigative methods.) NYSIIS is developing an improved system with the capability of matching lifted crime scene fingerprints with prints in the main criminal fingerprint file, utilizing a combination of computer search and microfilm retrieval technology.

Likewise, computer searching to identify perpetrators by personal appearance, modus operandi, trace data analysis, fraudulent check characteristics, etc., opens up an entire new spectrum of aids to criminal justice administration. In automatic license plate scanning for the apprehension of wanted motor vehicles a combination of optical and computer technologies has provided a viable solution to the epidemic stolen car problem. Most recently, through NYSIIS planning and research, it has been recognized that computerized organized crime intelligence systems hold the key to new opportunities for dealing with this nagging problem which, up to now, has been largely unresolved despite vast allocations of manpower resources.

Computer communications

The fantastic opportunities for criminal justice offered by computer technology depend to a very large extent upon compatible communications resources. Speedy computer processing of arrest fingerprint submissions

is pretty much in vain if the fingerprints require two or three days to arrive by mail and it takes two or three more days thereafter to receive the printout. The "magic" of computer identification of wanted vehicles passing on the highway is meaningless unless the "hit" message can be retrieved within a few seconds. The intra- and interstate exchange of identification and intelligence data must be facilitated by an entire array of computer compatible communications. Likewise, remote access and computer-to-computer interface depend upon appropriate telecommunications systems.

NYSIIS has seized the opportunities available to provide computer-related communications systems, both within New York State and on an interstate basis through its participation in SEARCH. A very significant development in this regard was the establishment of the first statewide facsimile network for the photo transmission of fingerprints from any point in the state to NYSIIS for computer processing and appropriate responses thereto by message facsimile. At the present time it still takes 14 minutes to transmit each set of fingerprints and an average of 4 minutes to respond with a criminal history record. These elapsed times are, of course, unsatisfactory and we are urgently pressing vendors to escalate their efforts to improve the technology.

In this connection, NYSIIS has been participating in the satellite transmission project of SEARCH, experimenting with the possible transmission of fingerprint card images via microwave and satellite rather than facsimile ground systems. Likewise, NYSIIS participated in the Project SEARCH interstate transmission of criminal history records which involved an advanced telecommunication network with remote access and computer-to-computer interface. NYSIIS and its many counterparts throughout the country are interfaced with the National Crime Information Center (NCIC) computer at the FBI in Washington, D.C., for purposes of stolen property and wanted identifications and most recently for the transmission of alphanumeric criminal history record data. Despite the sophisticated hardware presently available, we still need a number of breakthroughs in the area of communication technology in order to optimize the impact of the computer upon the criminal justice system.

Barren survival

Government agencies must justify their continued existence and make requests for their share of scarce resources each year. In the case of NYSIIS, it was necessary to convince the Legislature and its scrupulous

fiscal committee of the merits of this computerized criminal justice information system long before it produced any tangible benefits to anyone. Here again, the mystique of the computer provided opportunities to sustain interest in the glorious promise of a criminal justice information system.

A public information program which took full advantage of the public's fascination with computers and their incredible capabilities was mounted with significant success. Professional associations of police, district attorneys, correction, probation and parole officials provided loyal support during the barren years. They, too, were intrigued by the promised potential of computerized information sharing.

The continuous announcement of technological breakthroughs during development and the constant reiteration and reinforcement of the ultimate promise kept NYSIIS alive on the one hand; and, on the other hand, prevented the abortive development of unnecessary and redundant computer systems at the state and local levels. Many millions of dollars were saved by inhibiting the creation of systems which would duplicate what was already being planned on a more comprehensive basis and would perforce supplant any such truncated endeavors. The dazzle and the promise of the computer served NYSIIS well during the "lean" years of little production and much planning, research and development.

Security and privacy

From the very inception of NYSIIS it was evident that matters of security and privacy should be given prime attention. The same public awe of the computer which served NYSIIS so well in buying time for system analysis and development, could readily be changed to fear and turned against us. The computer compelled us to critically examine every facet of the planned structure to be certain that the system provided dynamic security and privacy, in order to equate in productive equilibrium the right of privacy and the need to share information.

NYSIIS recognized that we need to protect private personality as zealously as we protect private property. Long before the issue of the computer vs. privacy became a subject of national debate, NYSIIS had committed itself to a program of computer security which earned the praises of Congressman Gallagher, Oscar Ruebhausen, Orville Brim, Senator Ervin, Alan Westin, the New York Civil Liberties Union, the Vera Institute of Criminal Justice and many other persons and organizations who champion civil liberties.

As I testified recently before the Senate Subcommittee on Constitutional Rights:

"I firmly believe that computerized criminal justice information systems are essential for the effective administration of criminal justice and that such systems can be developed and operated with adequate security against unreasonable invasions of individual privacy—indeed, I believe that they can be so developed and operated as to provide new dimensions of personal freedom and protection for civil liberties and constitutional rights."

The concerns of NYSIIS were also the concerns of Project SEARCH and with their respective plans for protecting privacy and their voluntary adoption of stringent codes of ethics, we may rest assured that the computer sciences will remain alive and well in the criminal justice community.

SUMMARY

The future of criminal justice information systems, particularly at the state level, seems very bright indeed. The difficult years of development, which I have indicated by reference to the NYSIIS experience, are pretty much behind us. The miracle of mounting infusions

of money through welcome federal funding of computerized criminal identification bureaus and related technologies assures the fiscal support of such systems.

The very rapid and meaningful responses that computerized criminal justice information systems are providing for the felt needs of all functional branches of criminal justice administration are engendering professional support and commitment. The challenge of those who fear 1984, has been met head-on. We are leading the march for individual freedom and civil liberties, for the computer, properly controlled, is a willing slave to serve humanity, not a master of our fate.

Ultimately, computerized criminal justice information systems will be vindicated by two consummations:

1. Emergence of a coherent and coordinated system of criminal justice;
2. Reduction in the incidence of crime and effective apprehension, prosecution, adjudication and rehabilitation of offenders.

I am so bemused myself with the mystique of the computer that I sincerely believe the computer can accomplish this.

Automated court systems*

by RONALD L. BACA, MICHAEL G. CHAMBERS and WALTER L. PRINGLE

Symbiotics International Incorporated
Houston, Texas

and

STAYTON C. ROEHM

Harris County
Houston, Texas

INTRODUCTION

Why does our Judiciary continue to use antiquated methods in the courts instead of taking advantage of business automation techniques which have been so successfully utilized by private industry?

This paper answers this question and discusses some of the reasons why the courts, especially those in the larger cities, need such automation techniques.

The paper also describes what has been done in Houston, Texas, to solve this problem. The authors have worked closely with Harris County criminal justice officials for several years and have designed a completely automated criminal records system.

This system, called the Harris County Subject-in-Process Records System,¹ maintains pertinent information about criminal cases. This information is made available to the courts, law enforcement agencies, the District Attorney, and other agencies and departments involved with the judicial process.

JUDICIARY REQUIREMENTS

The court officials, especially in the larger cities, including judges, clerks of the courts and prosecuting attorneys, know what computers can do for them. Their conferences and professional publications constantly emphasize the importance of automation. They know also that somehow the processing of cases must be

* The development of the system described in this paper was financed in part by the Law Enforcement Assistance Administration with a grant awarded to Harris County, Texas, and administered by the Texas Criminal Justice Council.

speeded up or the wheels of justice will soon come to a grinding halt.

Chief Justice Warren E. Burger in his first state of the judiciary message in 1970 said: "In the supermarket age, we are like a merchant trying to operate a cracker barrel corner grocery store with the methods and equipment of 1900."

Litigants in criminal cases are experiencing delays of up to two years and more before their cases can even come to trial. This is particularly true in our larger metropolitan centers. After such a long period of time it is not unusual to find that witnesses involved in a case have moved away or even died. The standard solution to the delay problems is simply to add more courts. More courts mean more judges, more clerical support and more docketing problems.

Our courts are bogged down with manual book-keeping procedures. In many of the metropolitan areas it is not unusual to read about how someone was denied his freedom due to a simple clerical error or a breakdown in communications between the various departments that comprise the criminal justice system. Citizens often win judgments against law enforcement agencies in resulting litigation.

The problem of crimes committed by persons out on bond is a major one. Many states are implementing procedures to speed up the processing of cases involving dangerous persons who are free on bond. Such preferential treatment can result in more delays for innocent people who cannot post bond and must remain in jail.

It is suspected that a prime cause for much of the backlog and delay is due to a lack of coordination in docketing cases. Attorneys are often involved in a large

number of cases and therefore are frequently unavailable. It is also suspected that attorneys often ask for a postponement of one case in order to get a better setting for another case they are representing. These things are suspected, but without automation it is a formidable task to sift through the mountain of paperwork to determine bottlenecks in the judicial process and to formulate action to remove them.

Former Chief Justice Earl Warren, in a speech delivered at the annual meeting of the American Law Institute in 1966, said: "It seems to me there is a definite need for thorough analysis and study of the mechanics—in its physical aspects—of carrying on the business of the courts. I am led to this belief by the accomplishments of new data processing methods employed in other fields—medicine, for example."

Governmental agencies, especially on a local level, are quite inflexible in comparison to commercial businesses. Seemingly simple changes such as using an available computer facility to print an index of criminal defendants instead of manually entering each name in a "well-bound" journal often require amendments to state constitutions; or, at a minimum, require an interpretation by the State's Attorney General.

Of course, we are all too familiar with the problems posed by budgetary considerations and of officials who are not close enough to the problem and who find it difficult to approve expenditures for data processing.

Government often fails to use modern data processing procedures simply due to organizational restrictions. There is usually no one person or department to tie the various criminal justice departments and agencies together to organize and support the implementation of such a system.

What, then is being done to relieve our congested courts. The use of computers to streamline court procedures can presently be found in several large cities. Many of these systems, however, were implemented quickly to solve some immediate problems. What is desperately needed is a thorough analysis of the entire court system and the development of long range plans to solve the problems.

SYSTEM OBJECTIVES

The criminal justice officials in Harris County have long been aware of the administrative problems and have recently taken positive steps toward a solution by working together to develop what is now called the Harris County Subject-in-Process Records System. This computer system maintains all pertinent information about criminal cases and the defendants involved. The system information is available, via

printed reports and remote terminals, to the District Clerk, District Attorney, Sheriff, Probation Department, and the Courts.

The primary objectives in the design of the Harris County Subject-in-Process Records System were to produce a system which would provide an efficient means of monitoring the progress of criminal cases and to define methods of using such information to reduce the total time and effort required to process a case.

The system is designed in a manner to be mutually beneficial to the various County agencies and departments concerned with the criminal process. It is, whenever feasible and allowable under the statutes, designed to eliminate unnecessary duplication of records and effort amongst these agencies and departments.

Harris County records show that in 1966 the average time from indictment to trial was 18 months. Today the average is down to six months due to the diligent efforts of the County officials. U. S. Chief Justice Warren E. Burger, however, has urged that all criminal cases be brought to trial within 60 days of arrest.

ORGANIZATION AND DESIGN

The Harris County Subject-in-Process Records System was designed to eliminate the necessity of looking for information manually. Naturally, there are many manually processed legal documents. The computer system may, however, maintain copies of per-

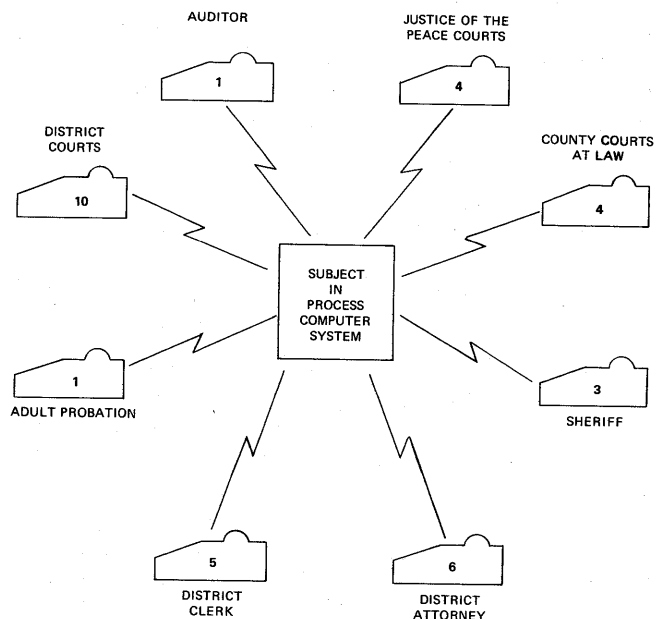


Figure 1—System interface

inent facts from each document and thereby provide instant response to many questions concerning criminal cases.

As a subject progresses from one step in the judicial process to the next, information regarding this progress is recorded in the computer system.

Figure 1 is a graphical representation showing which County departments interface with the System. The number depicted in each box indicates the number of terminals assigned to each department.

The Subject-in-Process Records System consists of teleprocessing and batch processing functions built around a nucleus of files serving as the System's data base. The System Organization flowchart shown in Figure 2 illustrates the system.² The various files and queues are shown in the center with the teleprocessing functions to the left and the batch processing functions to the right.

The three basic data files are the Case History File, Name and Identification Number File, and the Calendar File. These files are similar to those of the Basic Courts System³ (BCS) files, but several additions and modifications have been incorporated. The basic files are separated into active and inactive files to augment the on-line and batch oriented functions.

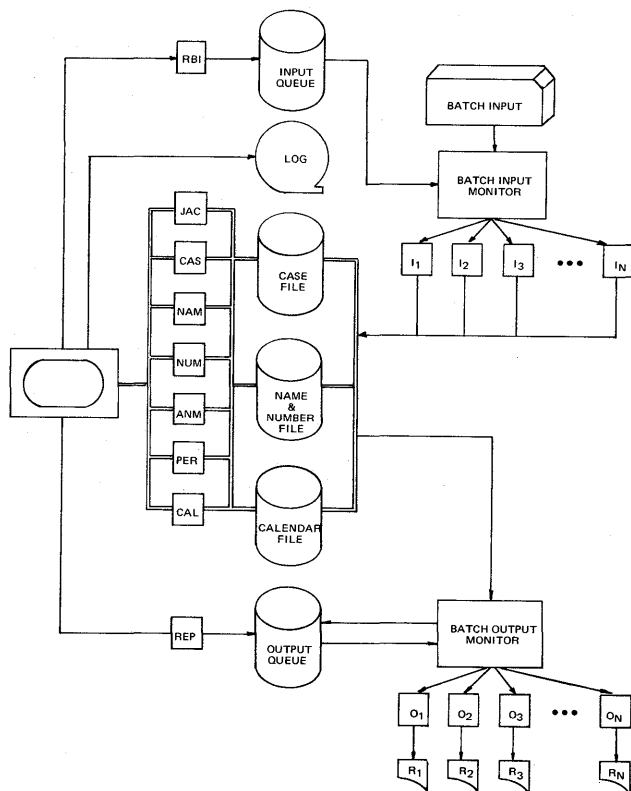


Figure 2—System organization

The remote terminal user has available to him nine basic teleprocessing functions. These consist of Remote Batch Input (RBI), Batch Output Reporting (REP) and seven on-line functions (CAS, NAM, NUM, ANM, PER, JAC and CAL) which aid the user in the interrogating, retrieving and updating of the basic data files via the remote terminals.

RBI allows for the input of batch data via the remote terminals by placing the input in a queue to be processed by the Batch Input Subsystem. REP allows the user to request batch output from the remote terminals by placing the requests on a queue to be processed by the Batch Output Subsystem. The seven on-line functions yield terminal displays to the terminal inquiries and are briefly described below:

- CAS: allows the user to search, retrieve and update the Case History File, and to display all associated transaction records at the terminal
- NAM: allows the user to search, retrieve and update the Name File and to display the desired records at the terminal
- NUM: allows the user to search, retrieve and update the Identification Number File and to display the desired records at the terminal
- ANM: allows the user to display all available identification numbers associated with a defendant to a case
- PER: allows the user to display all available personal descriptor information associated with a defendant
- JAC: allows the user to display the arrest/conviction history of a defendant
- CAL: allows the user to search, retrieve and update the Calendar File and to display the docket of a court

These teleprocessing functions are written in FASTER-LC⁴ and are incorporated into the system to augment the facility available to the user.

All terminal inquiries are logged on the Log File to provide system backup. In the event of a system failure, all transactions can be reconstructed and the integrity of the basic data files insured. The Log File also provides a data base for the analysis of user requests and overall terminal usage.

Batch Processing

The batch processing functions are divided into the Batch Input and Batch Output Subsystems. These subsystems are designed to interact with the queues built in the on-line mode and the basic data files. These

HARRIS COUNTY COMPLAINT INDEX - MONTH TO DATE JANUARY 22, 1970				PAGE 33
NUMBER	DEFENDANT'S NAME	OFFENSE CODE	OFFENSE DESCRIPTION	
2100-02	WASHINGTON SADIE	2501	FORGERY OF CHECKS	
2352-01	WATTS CHARLES R	3562	MARIJUANA - POSSESSING	
2260-01	WEST JAMES M	2270	BURG & THEFT	
2362-01	WHITE ROBERT	2300	THEFT BY BAILEE	

Figure 3—Complaint index

subsystems provide for the input of data to the files and the output of pre-defined system reports.

The Batch Input Monitor is a subsystem consisting of ANS COBOL programs which take the batch and remote batch input data and update the basic data files. This subsystem performs the necessary editing and formatting of the various data records and supplies diagnostic messages when appropriate.

The Batch Output Monitor is a subsystem consisting of ANS COBOL programs which queue the system requests for generating reports on pre-established frequencies. This subsystem also analyzes all system generated and user generated requests for batch output, eliminates duplication, establishes priorities and invokes the various batch output programs which produce the system reports.

The capabilities of the System include the ability to produce numerous printed reports at predetermined intervals or upon request. These reports include indexes, case histories, and summary reports.

The Complaint Index shown in Figure 3 is a list of all felony complaints which have been submitted to the Grand Jury. The index contains the defendant's name, a unique sequence number, the co-defendant suffix (a two-digit number used to identify defendants when there are more than one to a case), the offense code and the offense description. The Complaint Index is sorted

HARRIS COUNTY FELONY INDEX - MONTH TO DATE OCTOBER 25, 1970				PAGE 92
CASE NUMBER	DEFENDANT'S NAME	JUDGEMENT-RECORDS VOLUME PAGE	CASE DISPOSITION	
310154-02	ALLEN JOHN B	312 / 006	GUILTY	
308916-01	BOND JAMESON L			
309985-01	SMITH JOHN	310 / 125	NOT-GUILTY	
310225-03	WILLIAM WILLIAM W	311 / 205	NO BILLED	

Figure 4—Felony index

HARRIS COUNTY CASES PENDING THE GRAND JURY INDEX WEEK ENDING 12-18-70						PAGE 16
COMPLAINT NUMBER	DATE FILED	DEFENDANT'S NAME	J P COURT PREC POS	OFFENSE CODE	OFFENSE DESCRIPTION	
14296-01		WASHINGTON SADIE	2	2501	FORGERY OF CHECKS	
15113-02	08-10-70	WATTS CHARLES R	1 0	3562	MARIJUANA - POSSESSING	
14184-01		WEST JAMES M	1 1	2270	BURG & THEFT	
24780-01		WHITE ROBERT	2	2300	THEFT BY BAILEE	
*** TOTAL CASES PENDING THE GRAND JURY				4	***	
*** TOTAL CASES PENDING OVER 90 DAYS				1	***	

Figure 5—Grand jury index

and printed by defendant name and by the sequence number.

The Felony Index and the Misdemeanor Index are similar and contain the case number, co-defendant suffix, defendant's name, the volume and page of the judgment records and the case disposition. The indexes are printed in defendant name order. A sample is shown in Figure 4.

The Cases Pending the Grand Jury Index, see Figure 5, is an alphabetical list of all defendants of felony cases which have been bound over to the Grand Jury but have not been indicted or no-billed. The index contains the defendant's name, the complaint number with co-defendant suffix, the Justice of the Peace Court, and the offense. In addition, cases which have been pending the Grand Jury for 90 days or more are flagged by listing the date the case was filed in the Justice of the Peace Court.

The Cases Pending the District Courts Index and the Cases Pending the County Courts at Law Index consist of a list of cases which have been assigned to a County Court at Law or District Court, but are pending final disposition. The indexes are sorted in two major ways: by case number, and by ready status. A sample is shown in Figure 6.

The Case History Registers consist of chronological listings of all transactions concerning each case from the time the case number is issued until the case has been disposed of. A sample is shown in Figure 7.

HARRIS COUNTY FELONY CASES PENDING STATUS INDEX WEEK ENDING 12-04-70										PAGE 84
CASE NUMBER	C	T	DEFENDANT'S NAME	OFFENSE CODE	DEFENDANT STATUS	DEFENSE STATUS	PROSECUTION STATUS	IN PROCESS STATUS	CASE PENDING STATUS	
260571-01	161		ALLEN AL	3501	IN JAIL	READY	READY		READY FOR TRIAL	
261031-01	178		DOE JOHN A	2501	OUT ON BOND	READY	READY		READY FOR ARRAIGNMENT	
261825-01	178		PUBLIC JOHN Q	2301	WANTED	NOT READY	NOT READY		NOT READY TO CALENDAR	
262160-01	161		RAYMOND RAY C	2801	IN JAIL	READY	READY	TRIAL	CALENDAR	
*** TOTAL FELONY CASES PENDING								4	***	

Figure 6—District courts index

Teleprocessing

The System has the ability to display system information on remote terminal CRTs. As indicated earlier in Figure 2, which describes the organization of the data files, the system provides for the following terminal displays:

- NAME INDEX INQUIRY—NAM
- PERSONAL DESCRIPTOR INQUIRY—PER
- JAIL ARREST CONVICTION INQUIRY—JAC
- COURT CALENDAR INDEX INQUIRY—CAL
- IDENTIFYING NUMBER INDEX INQUIRY—NUM
- ASSOCIATED NUMBER INDEX INQUIRY—ANM
- CASE HISTORY REGISTER INQUIRY—CAS

The Name Index Inquiry, shown in detail in Figure 8, is a display which allows the user to identify information pertaining to persons involved in the judicial process. By supplying the System with the name of a person, the System responds by displaying all cases involving the person.

This inquiry capability is used to find the case number when only the name is known. The Name Index contains the names of all persons associated with complaints, misdemeanors, and felonies. That is, it contains the names of defendants, defense attorneys, prosecuting attorneys, witnesses, bondsmen, etc.

The Personal Descriptor Inquiry is used to answer requests for more identifying information about a defendant. Upon entering the defendant's name, the

Input:
 ▶ NAM, Smith, John, B.
 Response:

▶ NNN		NAME INDEX						
J000	LAST NAME	FIRST	M TI	CON C	CASE	MS CT	ENTITLEMENT	S FIL-DATE
J010	SMITH	JOHN	B SR	DEF 2	003945601	010	S V SMITH	06-29-70
J010	SMITH	JOHN	B SR	DEF 3	003816801	176	S V SMITH	05-18-70
J010	SMITH	JOHN	B SR	WIT 2	004137403	040	IMP THOMPkins	09-23-70

Figure 8—Name index inquiry

following information is provided:

- Place of Birth
- Date of Birth
- Height
- Weight
- Hair Color
- Ethnic Features
- Sex

The Jail Arrest Conviction Inquiry provides a display

Input:
 ▶ CAL, 08-24-70,176
 Response:

▶ SHN		CALENDAR							
R000	CAL-DATE	CNC	TIME	LOC					
R010	08-24-70	176	10:00	CR31					
S000	P C	CASE	MS REAS	PARTIES-DESCRIP	IN EST	ACT	DISP	FUT-DATE	PCC
S010	3	003456901	SENT	MICHAEL SMITH			1.0		
S020	3	003513201	TRY	JOHN A. DOE			8.0		
S030	3	003483601	SENT	JOHN Q. PUBLIC			.5		

Figure 9—Court calendar index inquiry

HARRIS COUNTY FELONY CASE HISTORY REGISTER WEEK ENDING 02-19-70		PAGE 107					
CASE NUMBER	DEFENDANT'S NAME	TRANSACTION TYPE	MINUTES	C	BAIL BOND	SET MADE	COMMENTS
14685-01	DOE JOHN L	INDICTMENT	12-27-69	001/213	183	500	THEFT
		CAPIAS ISS	12-28-69				
		CAPIAS RET	12-29-69				
		DISPOSITION	02-06-69	005/131			NON EXECUTABLE DEATH OF DEFENDANT
14686-01	SMITH HARRY B AKA ROBERTS HARRY	INDICTMENT	12-24-69	001/213	179	1000	MURDER
		CAPIAS ISS	12-24-69				
		CAPIAS RET	12-27-69				
		BOND MADE	12-30-69		1000		PLACED IN JAIL APPEARANCE BOND - ALLSTATE SURE CO
		HEARING	01-05-70				FOUND GANE AT TIME OF TRIAL
		PLEADING	01-29-70				PLEA OF NOT GUILTY
		JURY REQ	01-29-70				
		DISPOSITION	02-18-70	069/815			DECISION - GUILTY BY JURY PRISON - LIFE JAIL - 2 MONTHS
14687-01	BROWN HOLLY A	INDICTMENT	12-17-70	001/824	183	500	THEFT
		CAPIAS ISS	12-17-70				

Figure 7—Case history register

Input:
▶ CAS, 3,176-0028346-01

Response:

CASE HISTORY											
A000	C	CASE	MS	CT	ENTITLEMENT	DCPS	NEXT-APPEAR	FIL-DATE	CODE	LAST-CHG	PF
A010	3	002834601	176	S	V	DOE	B NN 11-15-70	08-26-70	2501	10-26-70	
B000 TRANSACTION -DATE- VOL/PAG IDENTIFIER VALUE INFORMATION											
B010	OFFENSE	082670				2501					FORGERY OF CHECKS
B030	COMPLAINT	082670	123/303	09-13-70		\$ 10000					004-0163104-01
B070	BOND MADE	082870				\$ 10000					
B130	INDICTMENT	102370				130					176-0028346-01
D000 PRINCIPAL JUDGMENT TRIAL DIS-ATT SHERIFF CLERK JURY LAW-LIB											
D010			100.00			15.00					5.00

Figure 10—Case history register inquiry

of all known arrest and conviction information concerning a particular defendant. This information may be used by the District Attorney's Office to prepare the prosecution and by the Sheriff's Office for criminal investigation purposes.

The Court Calendar Index Inquiry shown in Figure 9, allows the user to display the cases scheduled for a particular day in a particular court.

The Identifying Number Index Inquiry allows the user to identify a person and the cases that person is associated with by entering any one of several identifying numbers. The following identification numbers may be used:

- Complaint Number
- Sheriff's Number
- Texas Department of Public Safety Number
- FBI Number
- Social Security Number
- Operator License Number
- Arresting Agency Number
- Law Enforcement Number
- Grand Jury Records Section Number

This index allows the various agencies of the criminal justice process to communicate with the system by using their own identification numbers.

The Associated Number Index Inquiry allows the user to display all identification numbers associated with a person involved in the judicial process by supplying the system with a case number.

The Case History Register Inquiry shown in Figure

10 allows the user to display all transactions regarding a particular case.

SECURITY AND PRIVACY

In every computer system incorporating a large data base, security and privacy of information are important considerations. This is especially true in the case of a criminal records system. Two basic problems exist, errors and unauthorized access.

Errors are a result of mistakes occurring during the manual preparation of the input data. Errors on source documents, typographical or keypunching errors, and inadvertent omission of pertinent data are examples of the types of errors which can occur. The input routines detect invalid input data (numeric value out of range, alphabetic character in a numeric field, unknown code, etc.) and all data input via cards is verified by being displayed and matched with the source document. As data are input, routines also check for inconsistencies in data (e.g., a warrant for arrest is shown to be executed prior to being issued).

The second problem of unauthorized access to the data files is particularly critical. The criminal records system deals with highly sensitive information. Destruction or modification of this information would severely cripple the effective performance of criminal justice. Therefore, a considerable amount of effort has been made to ensure the integrity of the information contained in the criminal records system.

The criminal records system allows for the updating of records from remote terminals. This provides up-to-the-minute information in the files but can be a source of problems if unauthorized personnel have access to the terminals. Several steps have to be taken to alleviate this problem.

- Each person authorized to update the files is assigned an access code which is changed periodically. Without the code, modification of or additions to the files cannot occur. Furthermore, the access codes are valid only for a particular terminal.
- Certain terminals are designated as display terminals only and allow no modifications or additions to occur. In addition, those terminals which are allowed to make modifications or additions may be restricted to use only during those periods of the day when authorized persons are on duty.
- The system also provides file protection by terminals. Thus a particular terminal may be able to modify or add a record in the Name File but not in the Case or Calendar File.

- The system also has the ability to restrict the transactions allowed on a given terminal. Thus a particular terminal may be able to make an inquiry that is not allowed by some other terminal. This allows controls, via software, to be placed on the use of any terminal.

Periodically the information is transferred from disk storage to magnetic tape. Two copies of the files are made. One is stored locally and is used to recreate the files in the event of inadvertent (hardware malfunction) or deliberate destruction of the files currently recorded on disk storage. The other copy is kept at a remote location as protection against the destruction of both the files on disk storage and the magnetic tape copy.

While the above mentioned capabilities provide a means of protection, the ultimate success depends on the people involved and the extent to which the operating procedures are followed.

CONCLUSION

It should be noted that while this System was tailored specifically for Harris County, Texas, the concepts and design, if not some of the programs themselves, could be successfully applied to many other counties in Texas and throughout the country.

The System was designed with several important growth features in mind. Some of the possible additional capabilities being considered are simulation models which take advantage of the statistical information now available, a complete bookkeeping system for the Adult Probation Department to keep track of fines, supervisory fees and restitution payments, a complete jail record system from keeping track of personal effects and making cell assignments to computerized search capability of fingerprints and mugshots, and automated recording procedures for the Juvenile Probation Office.

The benefits of the Harris County Subject-in-Process

Records System are numerous. One of the primary benefits however, is the ability to obtain instantaneous response to a variety of questions concerning a case or a defendant. In the past, an inquirer was often transferred from one office to another as each office searched but failed to find the requested information.

Another benefit of primary importance is the System's ability to monitor the progress of each case and periodically report required actions. These action reports include lists of persons being held for no apparent reasons, cases that are ready for trial but have not been calendared and persons whose probation periods have elapsed but have not been officially terminated.

In addition to providing answers to questions and monitoring case progress, the System also provides numerous, written reports which assist the criminal justice officials in preparing a case for trial, scheduling each event of the trial, and preparing local and state statistical reports.

Another result of the computerized system is the ability to use the information to produce various statistical reports to aid in evaluating administrative procedures and to test hypothetical changes in these procedures. Additionally, quick access to accurate case load information is extremely useful for budget planning and evaluating future manpower and facility requirements.

All of these benefits aid significantly in reducing the time it takes to process a case.

REFERENCES

- 1 *The User's Manual*
Harris County subject-in-process records system
Symbiotics International Inc 1971
- 2 *Design Specifications*
Harris County subject-in-process records system
Symbiotics International Inc 1971
- 3 *Basic courts system (BCS)*
IBM form No GH20-0888 IBM Corp
- 4 *Faster-LC*
IBM form No SH20-0863 IBM Corp

Delphi and its potential impact on information systems

by MURRAY TUROFF

*Office of Emergency Preparedness, Executive Offices of the President
Washington, D. C.*

THE DELPHI METHOD^{1,2}

The Delphi method is basically defined as a method for the systematic solicitation and collation of informed judgments on a particular topic. The concept of "informed" here could mean poor people, if the subject were poverty, as well as the usual interpretation of "experts." The method has two important characteristics which distinguish it considerably from a polling procedure. The first is *feedback*, where the judgments of the individuals are collected, possibly formulated as a group response and fed back. Thus, each individual may view the results and consider whether he wishes to contribute more to the information and/or reconsider his earlier views. This round or phase structure may go through three to five iterations in the usual paper and pencil exercise. The second characteristic is that all responses are anonymous. The reasons for *anonymity* are much discussed in the literature and will not be reviewed here. However, there are circumstances where complete anonymity could be relaxed. In some cases it may be useful for the respondents to know who is participating in order to insure awareness that a peer group is involved in the discussion. Also, when a highly specialized subtopic enters the discussion it may be appropriate to permit an expert to endorse an item.

The primary objective of the Delphi process, as set forth in this paper, is the establishment of a "meaningful" group communication structure. If this view is accepted as correct, then the question of whether or not a Delphi exercise will produce "truth" is not a relevant one. The real issue, given the context of a particular problem, is what communication process or combination of processes will be most effective in terms of the resources available to examine the problem.

There appear to be five situations where the Delphi method clearly has an advantage over other alternatives:

- Where the individuals needed to contribute knowledge to the examination of a complex problem have

no history of adequate communication and the communication process must be structured to insure understanding;

- Where the problem is so broad that more individuals are needed than can meaningfully interact in a face-to-face exchange.
- Where disagreements among individuals are so severe that the communication process must be refereed.
- Where time is scarce for the individuals involved and/or geographical distances are large, thereby inhibiting frequent group meetings.
- Where a supplemental group communication process would be conducive to increasing the efficiency of the face-to-face meeting.

In order to emphasize the view that the Delphi is a communication process, Table I directly compares the properties of normal group communication modes and the non-automated and automated Delphi processes. The major differences lie in such areas as the ability of participants in a Delphi to interact with the group at their own convenience (i.e., random as opposed to co-incident), the ability to handle large groups, and the ability to structure the communication. With respect to time considerations, there is a certain degree of similarity between a Committee and a Delphi exercise since delays between meetings and rounds are unavoidable. Also, the Delphi Conference³⁻⁶ may be viewed conceptually as a random (occurring) conference call with a written record automatically produced. It is interesting to observe that within the context of the normal operation of these communication modes in the typical organization, governmental or industrial, the Delphi process appears to provide the individual with the greatest degree of individuality or freedom from restrictions on his expressions.

While the Table breaks down these systems separately, there is no reason why the examination of a particular problem would not be best served by a combination of these techniques. For example, a Delphi

TABLE I—Group Communication Techniques

	Conference Telephone Call	Committee Meeting	Formal Conference or Seminar	Delphi Exercise	Delphi Conference
Effective Group Size	Small	Small to Medium	Small to Large	Small to Large	Small to Large
Occurrence of Interaction by Individual	Coincident with Group	Coincident with Group	Coincident with Group	Random	Random
Length of Interaction	Short	Medium to Long	Long	Short to Medium	Short
Number of Interactions	Multiple, as required by group	Multiple, necessary time delays between	Single	Multiple, necessary time delays between	Multiple, as required by individual
Normal Mode Range	Equality to Chairman Control (Flexible)	Equality to Chairman Control (Flexible)	Presentation (Directed)	Equality to Monitor Control (Structured)	Equality to Monitor Control or Group Control and no Monitor (Structured)
Principle Costs	Communications	—Travel —Individuals time	—Travel —Individuals time —Fees	—Monitortime —Clerical —Secretarial	—Communications —Computer Usage
Other Characteristics	Time Urgent Considerations	Forced Delays		Forced Delays	Time Urgent Considerations
	—Equal flow of information to and from all —Can maximize psychological effects		—Efficient Flow of Information from few to many	—Equal flow of information to and from all —Can minimize psychological effects —Can minimize time demanded of respondents or conferees	

Conference may be used between committee meetings to arrive at an agenda and expose the areas of agreement and disagreement. This, in turn, would improve the efficiency of time spent in the actual committee meeting by focusing the discussion on those areas requiring review. In some instances this would also improve the efficiency of staff work before the meeting.

Usually a Delphi communication process, whether it be an exercise or conference undergoes four distinct phases. The first phase is usually characterized by exploration of the subject under discussion wherein each individual contributes additional information he feels is pertinent to the issue. The second phase usually involves the process of reaching an understanding of how the group views the issue (i.e., where they agree or disagree and what they mean by relative terms such as importance, desirability or feasibility). If there is significant disagreement, then that disagreement is explored in the next phase to bring out underlying reasons

for the differences and possibly to evaluate them. The last phase, a final evaluation, occurs when all previously gathered information has been initially evaluated and evaluations have been fed back for consideration.

The Delphi technique may be considered to have roots in the jury system and is, perhaps unfortunately, a rather simple idea. Because of this, many individuals have conducted one Delphi and only a few have gone on to do more than one. The process of designing a workable communication structure for a particular problem currently appears to be more an art than a science. However, a number of general reasons for failures have come to light from these less successful attempts:

- Utilizing a blank sheet of paper on the first round or phase and thereby implying that the respondents should waste their time in educating the design and monitor team;

- Poor techniques of summarizing and presenting the group response and insuring common interpretations of the evaluation scales utilized in the exercise;
- Ignoring and not exploring disagreements so that discouraged dissenters drop out and an artificial consensus is generated;
- Ignoring the fact that respondents to a Delphi are acting in a consultant mode in what may be a demanding exercise and should therefore be involved as a part of their normal job function or should receive normal consulting fees for participation.

The use of the Delphi process appears to have increased at an exponential rate over the past five years and on the surface seems incompatible with the limited amount of controlled experimentation that has taken place on the methodology itself. It is, however, meeting a demand for improved communications among larger and/or geographically dispersed groups which cannot be satisfied by other available techniques. It also serves the decision maker who wishes to seek out the potential secondary effects of a decision or policy which may involve a more diverse group of experts than is normally available. Also, technologists have become increasingly concerned that attempts to evaluate cost-benefit aspects through mathematical models often eliminate significant technical factors which they may feel are crucial criteria for the making of a decision. The Delphi process can, in this context, be viewed as an attempt to put human judgment, in terms of a group judgment by experts, on a par with a page of computer output. This is an unfortunate justification for the Delphi process, but from a pragmatic point of view it is a valid one in terms of decision processes in some organizations.

It can be expected that the use of Delphi will continue to grow. From this one can observe that a body of knowledge is developing on how to structure the human communication process for particular types or classes of problems. The abuse, as well as the use, of the technique is contributing to the development of this design methodology. It would seem obvious that any communication structure that employs pencil, paper, and the mails can, in principle, be duplicated in a real time mode on an interactive terminal-oriented computer-communication system. When this is done the resulting product is a continuous group communication process which eliminates some of the disadvantages in the paper and pencil type Delphi while retaining most advantages. It is the contention of this author that those in the computer field should begin to actively plagiarize the techniques of the Delphi design area for

building on-line conferencing systems tailored to various problem applications. The remainder of this paper attempts to support this assertion.

EXAMPLES*

In examining applications of the Delphi, one observes that the vast majority deal with forecasting the future. Because of this, many individuals associate the Delphi process solely with forecasting. However, in examining other Delphi exercises, one finds that they span a surprising diversity of applications:

- Examining the significance of historical events
- Gathering current and historical data
- Putting together the structure of a model
- Delineating the pros and cons associated with potential decision or policy options
- Developing causal relationships in complex economic or social phenomena
- Clarifying human interactions through role playing concepts.

If one adopts the view of Delphi as a communication tool, then this exhibited diversity of application is not surprising. A group communication process can, in theory, be applied to any problem area. The following will discuss some of these previous applications and indicate where they may lead in the future.

Dr. Williams of Johns Hopkins University has utilized the Delphi to obtain estimates of current rates of disease incidence and the success rate of various alternative treatments. Since hospital reports may reflect local reporting standards, there is considerable uncertainty associated with the data that is available. This phenomenon also occurs in other areas such as crime statistics. In applications of this sort, individuals are asked to supply low and high values as well as an explicit estimate. This type of exercise then proceeds in very much the form of a forecasting Delphi, although it deals with current data.

There are a surprising number of Delphi designers in the medical research and health care areas, some of these are Dr. A. Sheldon at Harvard, Dr. A. Bender at Smith Kline French, Dr. D. Gustafson at the University of Wisconsin, and Dr. G. Sideris, American Surgical Association.

A Recent Delphi on the Steel Industry⁷ by the National Materials Advisory Board of the National Research Council also attempted to gather estimates on

* See Reference 1 for explicit references to the examples mentioned.

the quantity of material flowing in and out of various processing segments of the industry. In such a case, even when a parameter is published it may only represent a percent of the industry. This percent factor may be only approximately known.

A proprietary Delphi was done which dealt only with historical events affecting the subject of the "Limitation or Elimination of Internal-Combustion Vehicles." Some eighty-two events were compiled by the respondent group and evaluated for explicit significance and "factors to watch" as a result of the events. The events were technological, economical, social, and political. The resulting summary arranging the events chronologically represented an excellent review and condensation for management. This same concept could easily be applied to a professional area and the computer field is perhaps overdue for a careful review of the literature. For example, it is doubtful that anyone in the field can claim he has read all that has been written on Management Information Systems. Probably all would agree, however, that the signal to noise ratio is small. It would be interesting to see the list of significant papers drawn up by a group of experts, and to discover how they would identify papers representing follow-on work to earlier papers and further developments that may occur as indicated by a particular paper. One added benefit of the Delphi is that an expert need not feel embarrassed to propose or argue for his own papers as significant. It is not clear, of course, that the group would always vote to include a suggested paper.

The concept of utilizing Delphi to examine history is a simple but powerful concept. Most organizations do not really do a good job on evaluating past performance and this often defeats the purpose of their planning efforts. The author hopes more applications of this type will be forthcoming.

Mr. S. Scheele of the SET, Inc. designed and executed a fascinating Delphi on the Role of Mentally Retarded in Society. Since he was dealing with a non-quantitatively oriented group, he relied very heavily on pictorial models which the individuals could fill in in order to represent human and societal interactions. Also inherent in the design were role playing concepts and a requirement for the respondents in answering different questions to assume different roles. This same concept applies to obtaining answers from individuals in political or public position where one would wish to ask for the individual's true view on an issue and the view he would espouse if required to take a public position.

The role playing concept in the Delphi has implications for an organization in the sense that most budget allocation procedures may be viewed as a form of polling where each manager submits his requests to a central source. When budget cuts must be made,

there is a great deal of competition among the divisional groups, often resulting in antagonism and a complete breakdown of lateral cooperation and communication. The budget process could be "carefully" recast in a Delphi mode and each manager asked to assume the roles of other managers and to attempt justification of budget segments other than his own. This could lead to more understanding of the final allocation for all concerned and correspondingly less antagonism. The validity of the above general suggestion is, however, extremely dependent upon the particular organization and details of the environment, operation, and makeup.

Norman Dalkey's "Quality of Life" Delphi is a classic simple example of utilizing a Delphi to obtain subjective evaluations which could not be gained by any analytic method. Here the respondents were asked to itemize and define a set of variables which comprised the Quality of Life and were measurable in at least an empirical sense. The feedback mechanism was necessary to arrive at mutually understandable definitions and the anonymity was desirable to avoid the embarrassment of individuals who might rate factors such as "aggression" higher than the group as a whole. The same type of Delphi was conducted on a group of corporate executives to determine if their ranking of the Quality of Life variables corresponded to the corporation executive benefit program.

Many individuals have a mistaken impression that consensus is a goal of all Delphi Exercises. When exploring policy or decision issues, the goal may be to develop the strongest set of pros and cons concerning a given issue. In a sense then, some policy Delphis seek to at least explore disagreement if not to directly foster it through the makeup of the respondent group. Even if a decision maker has reached a view on an issue, it may be of interest to him to seek out the opposing view to be forewarned of difficulties he may encounter when his decision is made public. The discovery of a consensus among opposing advocates on underlying issues or compromise positions may make the exercise doubly useful but may not be the primary goal.

In the steel Delphi mentioned earlier, the respondents were given a flow model diagram of steel processing which was intended to collect data on the flow of material in each path. The initial model was put together by an expert. However, many of the respondents to the exercise decided that the diagram was not sufficient to express what they felt were significant connections. As a result of the uninvited modification of the model, the diagram obtained after two Delphi rounds was considerably more detailed and realistic. This leads to the proposition that Delphi can be utilized to build model structures for complex processes. The difficulty with some of the plans for designing computer

graphic systems for group engineering design efforts is that the computer people often forget that the concept can be first tried with pencil and paper on a real-life problem to see if a workable communication structure would result. If it succeeds in a Delphi Exercise mode then there is a higher probability of success in the automated version. In many ways, the Delphi activity as it occurs today is conducting a significant experimentation program for the field of computer sciences. This fact appears to have, thus far, escaped the notice of most computer personnel. The general concept of pre-testing an information system design by paper and pencil exercise before it is frozen in the concrete of our "flexible" computer system deserves more attention than it has received.

One very significant aspect of the Delphi area has been the design of attempts to discover views on causal relationships underlying complex physical, social, and/or economic systems. While many design techniques have been tested, one in particular has gained wide use because of the ease with which even non-quantitatively oriented individuals can supply answers. This communication format is generally referred to as "Cross Impact"⁸ and involves a matrix formulation of causal effects where the user is asked to supply either probabilities, odds, or weights depending on the particulars of the formalism. While the approach is easy to use, the analysis of the results is less clear because one is asking only for a small, but feasible portion of the information required to rigorously specify the problem and therefore consistency checks can only be approximations. At least four different methods of analysis are currently being used. An important difference for some of these approaches is the ease with which the method can be incorporated into an interactive mode on a computer system. In experiments the author has conducted with a method of treatment suited for a computer, one finds that a non-programming user, by supplying answers to a cross impact form, can in effect build his own model of the future which he can then subject to perturbations to see the effects of alternative decisions or policy. This becomes very useful as an aid to the thinking through of a complex situation. The interactive feature is extremely important in allowing an individual to modify his initial estimates until he feels he has obtained consistency between these and the inferences provided by the analytic treatment. Once a user is satisfied with the estimates obtained in this one-person game mode, they may be applied automatically to the formation of a group estimate and may allow individuals to see the differences in judgment that may occur for both the magnitude and the direction of the causal effects. This process quickly focuses the group's attention on areas of either disagreement or uncertainty

which then may be discussed in a committee process or a general discussion-oriented Delphi.

The particular utility of the cross impact formalism in a planning environment will become evident in the next section.

In terms of the author's knowledge alone, there are at least thirty distinctive Delphi designs which have been successfully applied to particular problem areas. Each one of these is a potential candidate for automation on a terminal-oriented computer system in order to implement a real time conference system. While many of these require graphical input, a sizable number can be implemented utilizing the common teletype terminal. When the computer is introduced we also introduce the ability to provide for the Delphi respondent both analytical tools and selective data bases which he may utilize to sharpen his judgments before they are contributed to the group response.

A significant observable effect of a computerized conference system is the group pressure to restrict discussion to the meat of the issue. Verbose statements always tend to receive low acceptance votes and individuals quickly learn, because of this, to sharpen their position if they wish to make a point.

Putting all these factors together with the real time nature of such a system, we can begin to visualize the results as approaching something that might be termed a "collective human intelligence" capability. In terms of the current state of the art in the computer field there may be a great deal more pay-off in easing the ability of humans to contribute the intelligence to the computer than in attempting to get the computer to simulate intelligence.

INFORMATION SYSTEMS

In most organizations today, the individuals or groups involved in forecasting and/or planning* usually exhibit the greatest desire to foster lateral communication. This often comes from a realization that uncertainties and ancillary considerations must be carefully explored if the organization is to avoid problems in the future. The desire to seek out the specialists in the organization regardless of where they sit, combined with the requirement to minimize the time they must give up from their normal functions, has led to an increasing use of the Delphi by the forecasting groups.

* The exception to this generality occurs when there is a belief that planning or forecasting can be reduced to only the consideration of dollars and alternative dollar equivalents or investments. Perhaps more organizations take this view than is warranted by their situation.

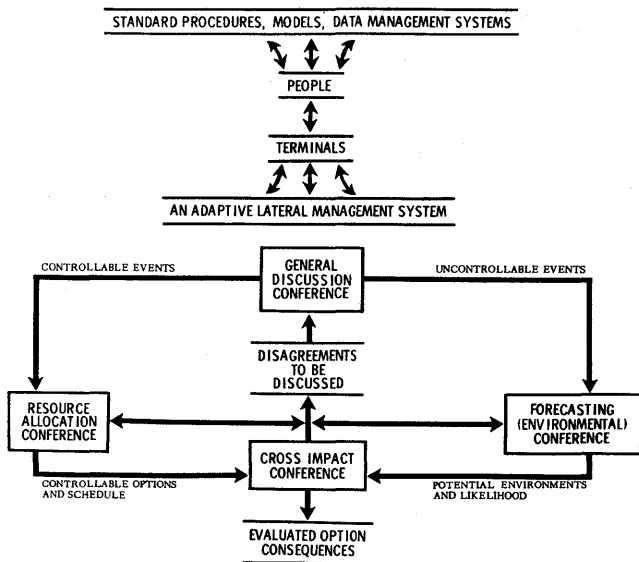


Figure 1

Due to the increasingly complex environment that most organizations face today, a similar circumstance has developed with respect to day-to-day management functions. The need for committee participation is beginning to make heavy demands upon the time of many managers. While the paper and pencil Delphi process has been introduced in some cases to alleviate the situation, it does not always meet the time urgent requirements associated with some management activities. These seem, therefore, to be a rapidly increasing interest in implementing more efficient communication techniques to deal with complex management problems. The automated Delphi or Delphi Conferencing may very well be the answer to this problem. In fact, one can conceptually lay out a highly adaptive Management Information System based upon this view.

Given that the organization has a problem to be examined and resolved, the first step is to pinpoint the individuals who can contribute to the process independent of their organizational or geographical location. They, as individuals, may contribute via terminals at their convenience to a general discussion conference (see Figure 1).

Requests for information on the potential environment (i.e., those factors not under control of the organization) will emerge from this discussion. These requests are shifted to a specialized conference structure which may involve only a subset of the general conference group and other specialists as needed. The communication structure for this forecasting conference is probably typical of many of the forecasting Delphis already in existence.

Potential program options would also evolve from the general discussion conference. These options would be shifted to another secondary or specialized conference to evaluate questions of resource allocation within the organization. This type of conference would possibly have various analytical support routines involving optimal allocation of resources among combinations of program options.

In both the resource allocation and forecasting conferences one would expect uncertainties or disagreements to occur which should be fed back to the general discussion conference for resolution. The results of these efforts would be a set of program options and potential environments which may now be played off, one against the other, in a conference structured along the lines of a 'Cross Impact' exercise.

In this third conference, additional uncertainties and disagreements may arise to be fed back to the general discussion conference. It is also possible, if not likely, that the results of the cross impact may trigger the requirement to introduce new program options or to examine a newly introduced aspect of the environment. One then views the interaction of these four conference structures as a continuous communication and feedback structure.

This basic set of four conferences may be replicated for each problem the organization wishes to put through this process. Therefore any one individual may be involved in a number of different problems. Also, a particular problem may be perpetual in nature so that the activity never stops but different individuals may enter and leave the discussion as a need for their particular speciality arises and is satisfied. The result of this is a highly adaptive and flexible structure for problem solving which the author feels exhibits all the characteristics of a Management Information System or what MIS should be.

The underlying premise behind the adoption of such a system is that while the organization may have elaborate data management systems and simulations or models, there are no algorithms allowing the data flowing through the normal organization procedure to automatically be transformed into a form directly suitable for addressing management problems as they occur. The view here is that individuals provide the best available mechanism for discriminating, reorganizing and presenting the portion of the data needed for problem consideration.

The problem that appears to exist in many current MIS efforts is the view that it is possible to introduce automation to the point where the person at the top can press a few keys on the terminal and all the pertinent data in a form appropriate to his problem will be retrieved. This is true only to the extent one believes

that all the problems that will be considered can be predefined.

Most of the current MIS efforts are based upon what the philosophy of science people would term a 'Leibnizian Inquiring System'⁹ where the approach is to believe that one can construct a model of a physical process independent of the data inputs. This view underlies the mathematical and physical sciences, and the attempt of the soft sciences, including the management sciences, to emulate this philosophy has perhaps created some of the problems in applying work in this area to real problems.

It is of interest to note that any particular Delphi design, or communication structure can be characterized in terms of one of the Inquiring Systems specified in Churchman's writings. However, very few Delphis fall into the category of being Leibnizian since there is usually a basic recognition in Delphi structures that the problem and data are inseparable. The policy type Delphi can, for example, be characterized as a "Hegelian Inquirer" which in its extreme assumes that any particular data, through its representation, can be used to support contrary positions. Information is considered fundamentally, in this view, as a property of the conflict between contesting points of view.

The lateral management system that this paper has attempted to describe can be viewed as what has been termed a "Singer-Churchmanian Inquirer" where it is assumed that "there are a multiplicity of models, theories, and inquirers for looking at the world, no one of which has *absolute* priority over the other". With this view, one is quickly forced to the position that each new problem must be examined within the context of the available information and potential analysis tools in order to arrive at a treatment. Therefore, we must utilize the only information processor which can evaluate* among alternative approaches—humans.

When one realizes that a majority of the efforts associated with trying to apply computer systems to the problems facing organizations are based upon a Leibnizian Inquiring philosophy, then one of the little known, but crucial dangers associated with computer systems becomes clear. Organizations are forced, by both accounting requirements and command (i.e., focusing of responsibility) requirements, into adopting a hierarchical structure. Because the environment confronting these organizations has become increasingly complex the resulting structure does not often match the problems that arise. The usual Leibnizian reaction to this situation is to reorganize so that a new structure fitting the problem emerges. Because structures, es-

pecially if they retain the hierarchical property, are fairly rigid, and the situation today is characterized by numerous problems wherein no one structure is common to all, these attempts to reorganize do not usually accomplish the desired goal.

Since organizations are made up of at least a subset of intelligent human beings, the inadequacies of the organizational structure and the resulting established communication channels are at least obvious to some. The result is a growing lack in many organizations of effective communications about various problems. The individual perceiving the situation faces a choice of either establishing informal communication channels and perhaps suffering consequences for bypassing the established modes or suffering in silence and adapting a game-playing attitude toward the communication process available to him. When this latter attitude is characteristic of a large segment of the organization, there is no longer an effective human communication process and individuals become extremely unresponsive to attempts to effectively deal with problems. This is further complicated in times of tight budgets where there is competition for resources among different segments of the organization.

Given the above situation in an organization, what happens when a computerized Management Information System designed along Leibnizian lines is introduced? A well designed system of this sort gives the illusion of intelligence by being very responsive to the individuals communicating with it. Since it is data independent it can translate any input data provided by the humans into an apparently original output or consequence. Psychologists would possibly agree that given the alternative of an unresponsive human communication process or a responsive man-machine communication process most individuals will shift their efforts at communication to the machine. We then find the computer becoming a surrogate for a repressed individual desire for effective communication. In addition, since the Leibnizian view of the world appears invalid for the type of problems confronting most organizations, then the introduction of a Management Information System based on this concept is a form of deception. The result is that the human is still playing a game, although with the computer he may be less aware the existence of the game than in the process of dealing with humans.

What the author therefore believes to be a real danger of computer usage over the long term is the ability of these systems to subjugate the desire to treat problems associated with human communications in organization by providing an image that an effective communication process exists. There is the possibility of a world ten to twenty years hence where a majority

* In the sense of applying value to the alternative approaches.

of the professional populace believes it is performing a useful function, but is, in fact, engaged in a game from which no tangible benefits result.

USER REQUIREMENTS FOR CONFERENCING¹⁰

The first and paramount requirement is that the designer of a conference structure have available a user oriented language (i.e., BASIC, JOSS, APL, TINT, etc.) in which the conference can be programmed. The general rule about conferencing systems is that any group of users will, through experience, have a considerable number of modifications to make. Also, it can be expected that a new type of problem may dictate a new communication structure. The role of the computer specialist should be to provide those features in a user language and machine executive which will allow the designer the flexibility of programming communication structures which may be intertwined with simple or complex analytical expressions (i.e., from vote averaging to optimization models). The basic system requirements are twofold and very similar to those required for on-line simulations involving a group of humans:

- (1) Simultaneous attempts by two or more individuals at separate terminals to write in the same file should not cause garbling of the file.
- (2) Errors in input or noise on the line should not confuse the user by throwing him out of the program and into the compiler or executive program.

There are several ways to meet the requirements. Summarized here are the particular features available in XBASIC* on the UNIVAC 1108 that allow the writing of conferencing systems.

An XBASIC program can execute a subset of the executive level commands on the 1108. This capability is helpful in allowing the program to assign the common file exclusively (using the executive command) for a short time (less than a second) to the conferee who is inputting data at that moment. The program then frees the file for any other conferees desiring to write in it. This simple feature solves the first requirement.

In order for the interaction program to do all the error checking, a full capability for decoding strings is required. Everything (even a number) entered via the terminal or via noise on the communication line is read as a string and checked for allowed choices. Therefore,

* Proprietary processor developed by Language and Systems Development Inc.

the ability to accomplish string manipulation and provide storage of string variables is required.

Output, especially for non-programmers, must be neat. Therefore, format or form control, such as is provided in FORTRAN or JOSS, for example, must be part of the language.

A good test of the sufficiency of the string handling capabilities in a user language is provided by examining the difficulty of writing one of the standard interactive text editors in the user language.

Although the above items are sufficient, a number of other features will make things easier or more efficient. Many of these are covered in Hall's paper in the 1971 SJCC proceedings.⁴

Many computer professionals appear to have believed until now that any user can be placed in one of five categories. The user does calculations, or he looks at data, or he manipulates strings, or he edits text, or he files things away; but he always does just one of these things, and the system capabilities are slanted accordingly. All the users I have ever encountered seem to do all the above in their daily non-computer chores. It is time for user languages to reflect a more realistic picture of users and their requirements.

If computer conferencing is to be a successful operation, the design and modification of conference structures with respect to the dictates of the problem being examined must be largely carried out by the users. Once a successful structure has evolved, a good systems programmer will have a role in making the overall operation efficient, provided the system is to receive long term use.

REFERENCES

- 1 A more detailed discussion of the Delphi and a comprehensive bibliography of this area may be found in *The Design of a Policy Delphi* by Murray Turoff, *Journal of Technological Forecasting and Social Change*, Vol. 2, No. 2, 1970.
- 2 A comparison of Delphi as a planning tool with other planning tools may be found in *Technological Forecasting and Engineering Materials* by the Committee on Technological Forecasting of the National Materials Advisory Board of the National Research Council, NMAB-279, December 1970.
- 3 The history of a particular Delphi Conference application may be found in *Delphi Conferencing (i.e., Computer Based Conferencing with Anonymity)* by Murray Turoff, *Journal of Technological Forecasting and Social Change*, Vol. 3, No. 2, 1971 (publisher: American Elsevier). This paper contains the complete design of the user interaction.
- 4 Details on implementing the above conference system on a computer may be found in *Implementation of an Interactive Conference System* by Thomas W. Hall, *Proceedings of the 1971 Spring Joint Computer Conference*.

- 5 An abbreviated report on this topic may be found in *Industrial Applications of Technological Forecasting and its use in R&D Management*, Wiley 1971, edited by M. Cetron and C. Ralph.
- 6 An explanation of the Delphi Conferencing concept for the layman is available in the April 1971 issue of the *Futurist* (magazine of the World Future Society, Washington, D. C.).
- 7 Two other large recent Delphis (involving 40 to 100 experts) reviewing the potential future of an industrial sector was one on computers by IBM and one on the Housing Industry by Selwyn Enzer of the Institute for the Future. See *Some Prospects for Residential Housing by 1985*, IFF report R-13, January 1971. The "Delphi Exploration of the Ferroalloy and Steel Industry" should be available from NMAB in late 71 and contains a detailed history of the effort involved in carrying out a large scale Delphi exercise.
- 8 For a review of this literature see *An Alternative Approach to Cross Impact Analyses* by Murray Turoff, Submitted to the *Journal of Technological Forecasting* for publication early 1972. This paper also illustrates the use of Cross Impact in an information system context.
- 9 See *What is Information? A Philosophical Analysis* by Jan J. Mitroff, Interdisciplinary Program in Information Sciences, University of Pittsburgh (to be published). Also the writings on Inquiring Systems by C. West Churchman (Internal working papers 28, 29, 45, 46, 49 on Inquiring Systems, Space Sciences Lab., University of California Berkeley, to be compiled in a book).
- 10 The author has discussed some of these issues earlier: *Immediate Access and the User*, *Datamation*, August 1966 and *Immediate Access and the User Revisited*, *Datamation* May 1969.

OTHER REFERENCE MATERIAL

Most of the current literature on Delphi appears in the *Journal of Technological Forecasting and Social Change*, *Futures*, or the *Futurist* (the magazine of the World Future Society). The World Future Society also runs a supplemental bulletin in which on-going work is reported usually long before publication.

The Institute for the Future is doing continuing work on applying the Delphi to fairly complex problems. Their reports and working papers should be of considerable interest to anyone planning to utilize the technique on a large scale problem.

Norman Dalkey at RAND has been carrying on a continuing series of experiments on the methodology and many of his recent papers are mandatory reading for potential practitioners.

The following items are meant to augment the extensive bibliography already available in the paper: *The Design of a Policy Delphi*.

A recent OEO report discusses the role of Delphi Conferencing as a component of an "Executive Infor-

mation System", for the Governor of Wisconsin:

- *GENIE (Government Executives Normative Information Expediter)* by D. Sam Scheele*, Vincent De Sante and Edward Glasser, March 1971.

A version of the Delphi Conferencing System has been implemented in TRAC on the PDP-10 by Claude Kagan of Western Electric Research, Princeton, N.J..

The proceedings of the First General Assembly of the World Future Society (held in May of 1971 in Washington, D.C., and to be published late 1971 or early 1972) contain two papers of interest:

- *On the Design of Inquiring System—A Guide to Information Systems of the Future* by Ian I. Mitroff.
- *Three-Hundred and Seventy-Third Meeting of the Council on Social and Economic Cybernetic Stability in the Year 2011* by Murray Turoff.

The first provides a review and literature guide to the concept of "Inquiring Systems" and the second is a forecasting scenario which carries some current tendencies in the computer field to their dangerous, but perhaps logical, extreme.

Prof. Mitroff also has a paper in Vol. 17, No. 10, June 1971 issue of *Management Science* which deals with a particular application of a Hegelian Inquirer:

- *A Communication Model of Dialectal Inquiring Systems—A Strategy for Strategic Planning*.

The 1971 IFORS (International Federation of Operations Research Societies) meeting on Cost Effectiveness held in May of 1971 in Washington, D.C., had a working session on Delphi. (The proceedings should be published in 1972 by Wiley.) The report on the working session provides a synopsis on the Delphi method and also reports on an experiment held in the session where the audience voted (as to agreement or disagreement) with respect to twenty-one conclusions contained in a technical presentation. The vote was taken both before and after the presentation. This modified form of Delphi provided a clear measure of the effectiveness of the presentation and its utility as an educational experience for the audience. One cannot help but conjecture that extensive use of this technique at professional meetings might either significantly decrease the number of papers

* Mr. Scheele (Social Engineering Technology Inc., L.A.) presented this concept at a session in the SJCC 1971; it is not, however, available in those proceedings.

submitted or significantly improve the quality of the presentations.

The IFORS proceedings also contain a review article on Multidimensional Scaling and its potential use in Delphis by J. Douglas Carroll of Bell Telephone Laboratories. Work of this sort in the field of psychology is pertinent to using the Delphi for obtaining value judgments.

Those interested in the use of Delphi in social indicators should examine:

- *Experimental Assessment of Delphi Procedures with Group Value Judgments* by Norman Dalkey and Daniel Rourke, RAND Report R-6-12-ARPA, February 1971.

Two recent attempts to validate Delphi exercises with respect to "real" applications were carried out by Dr. John W. Williamson of the School of Hygiene and Public Health, Johns Hopkins University. These were:

- *Prognostic Epidemiology of Breast Cancer and Prognostic Epidemiology of Absenteeism*

A recent Delphi of interest to the computer field is:

- *A Delphi Inquiry into the Future Economic Risks of Computer-Based Systems* Institute for the Future, Middletown, Connecticut.

There is considerable activity in the use or potential use of Delphi in the area of regional or urban planning. An example of this may be found in:

- *Sea Grant Delphi Exercises: Techniques for Utilizing Informed Judgments of a Multi-Disciplinary Team of Researchers*, by John D. Ludlow, Bureau of Business Research, University of Michigan, Working Paper 22, Jan. 1971.

A number of county governments are utilizing the Delphi technique internally.

The Delphi literature has become quite rich in recent years with respect to the diversity of applications. One can quite easily be amazed at the number of information systems being designed and utilized without the use of a computer. This is especially significant if one concludes, as I have, that many of these designs are closer to meeting MIS requirements than other activities designed on computers and labeled as MIS.

Technology for group dialogue and social choice*

by THOMAS B. SHERIDAN

The Massachusetts Institute of Technology
Cambridge, Massachusetts

INTRODUCTION

Usually the best way to discuss and resolve the choices that arise within groups of people is face-to-face and personally. For this reason, city planners and educators alike are calling for new kinds of communities for working, living, and learning, based more on familial relationships between people than on contractual relationships. When people get to know one another, conflicts have a way of being accommodated.

Beyond the circle of intimacy the problem of communication is obviously much greater; and while social issues can still be resolved more or less arbitrarily, it is more difficult to resolve them satisfactorily.

The "circle of intimacy" is constrained in its radius. One analyst has estimated that the average person in his lifetime can get to know, on a personal, face-to-face basis, only about 700 people—and surely one can know well only a much smaller number. The precise number is not important: the point is that it is dictated by the limitations of human behavior and is not greatly affected by urban population growth, by speed of transportation and communication, by affluence, or by any other technologically induced change in the human condition.

Indeed, these changes underlie the problem as we know it. Although the number of people with whom we have intimate face-to-face communication during a lifetime remains constant, we are in close proximity to more and more people.

We are, moreover, a great deal more dependent on one another than we used to be when American society was largely agrarian. We are all committed together in planning and paying for highways and welfare. We pollute each other's water and air. We share the risks

and the costs of our military-industrial complex and the foreign policy which it serves. Technology, while aggravating the selfishly independent consumption of common resources, has made communications beyond the circle of intimacy both more awkward and more urgent.

Beyond the circle of intimacy, what kind of communications make sense? Surely most of us do not demand personal interactions with "all those other people." Yet in order to participate realistically in the decisions of industry and commerce, and in government programs to aid and regulate the processes which affect us intimately, we as citizens need to communicate with and understand the whole cross-section of other citizens.

Does technology help us in this? Can it help us do it better? We may now dial on the telephone practically anywhere in the world, to hear and be heard with relatively high fidelity and convenience. We may watch on our television sets news as it breaks around the world and observe our President as though he were in our living room. We can communicate individually with great flexibility; and at our own convenience we can be spectators en masse to important events.

But effective governance in a democracy requires more than this. It requires that citizens, in various ways and with respect to various public issues, can make their preferences known quickly and conveniently to those in power. We now have available two obvious channels for such "citizen feedback." First, we go to the polls roughly once a year and vote for a slate of candidates; second, we write letters to our elected representatives.

There are other channels by which we make our feelings known, of course—by purchasing power, by protest, etc. But the average citizen wields relatively little influence on his government in these latter ways. In terms of effective information transmitted per unit time, none of the presently available channels of citizen feedback rivals the flow from the centers of power outward to the citizens via television and the press.

* The research at M.I.T. described herein is supported on National Science Foundation Grant GT-16, "Citizen Feedback and Opinion Formulation" and a project "Citizen Involvement in Setting Goals for Education in Massachusetts" with the Massachusetts Department of Education.

What is it that stands in the way of using technology for greater public participation in the important compromise decisions of government, such as whether we build a certain weapon, or an S.S.T., or what taxes we should pay to fund what federal program, or where the law should draw the line which may limit one person's freedom in order to maintain that of others?

Somehow in an earlier day decisions were simpler and could involve fewer people—especially when it came to the use of technology. If the problem was to span a river and if materials and the skills were available, you went ahead and built the bridge. It would be good for everyone. Thus with other blessings of technology. There seemed little question that higher capacity machines of production or more sophisticated weapons were inherently better. There seemed to be an infinite supply of air, water, land, minerals, and energy. Today, by contrast, every modern government policy decision is in effect a compromise—and the advantages and disadvantages have to be weighed not only in terms of their benefits and costs for the present clientele, but also for future generations. We are interdependent not only in space but in time.

Such complex resource allocation and benefit-cost problems have been attacked by the whole gamut of mathematical and simulation tools of operations research. But these "objective" techniques ultimately depend upon subjective value criteria—which are valid only so far as there are effective communication procedures by which people can specify their values in useful form.

THE FORMAL SOCIAL CHOICE PROBLEM

The long-run prospects are bright, I think, that new technology can play a major role in bringing the citizenry together; individually or in small groups, communicating and participating in decisions, not only to help the decision makers but also for the purpose of educating themselves and each other. Hardware in itself is not the principal hurdle. No new breakthroughs are required. What is needed, rather, is a concerted effort in applying present technology to a very classical problem of economics and politics called "social choice"—the problem of how two or more people can communicate, compare values or preferences on a common scale, and come to a common judgment or preference ordering.

Even when we are brought together in a meeting room it is often very awkward to carry on meaningful communication due to lack of shared assumptions, fear of losing anonymity or fear of seeming inarticulate, etc. Therefore, a few excitable or most articulate

persons may have the floor to themselves while others, who have equally intense feelings or depth of knowledge on the subject, may go away from the meeting having had little or no influence.

It is when we consider the electronic digital computer that the major contributions of technology to social choice and citizen feedback are foreseen. Given the computer, with a relatively simple independent data channel to each participant, one can collect individual responses from all participants and show anyone the important features of the aggregate—and do this, for practical purposes, instantaneously.

Much of technology for such a system exists today. What is needed is thoughtful design—with emphasis on how the machine and the people interact: the way questions are posed to the group participants; the design of response languages which are flexible enough so that each participant can "say" (encode) his reaction to a given question in that language, yet simple enough for the computer to read and analyze; and the design of displays which show the "interesting features" or "pertinent statistics" of the response data aggregate.

This task will require an admixture of experimental psychology and systems engineering. It will be highly empirical, in the same way that the related field of computer-aided learning is highly empirical.

The central question is, how can we establish scales of value which are mutually commensurable among different people? Many of the ancient philosophers wrote about this problem. The Englishmen Jeremy Bentham and John Stuart Mill first developed the idea of "utility" as a yardstick which could compare different kinds of things and events for the *same person*. More recently the American mathematician Von Neumann added the idea that not only is the worth of an event proportional to its utility, but that of an unanticipated event is proportional also to the probability that it will happen.¹ This simple idea created a giant step in mathematically evaluating combinations of events with differing utilities and differing probabilities—but again for a single person.

The recent history of comparing values for *different people* has been a discouraging one—primarily because of a landmark contribution by economist Kenneth Arrow.² He showed that, if you know how each of a set of individuals orders his preferences among alternatives, there is no procedure which is fair and will always work by which, from this data, the group as a whole may order its preferences (i.e., determine a "social choice"). In essence he made four seemingly fair and reasonable assumptions: (1) the social ordering of preferences is to be based on the individual orderings; (2) there is no "dictator" whom everyone imitates; (3) if every individual prefers alternative A to alternative B, the

society will also prefer A to B; and, (4) if A and B are on the list of alternatives to be ordered, it is irrelevant how people feel about some alternative C, which is not on the list, relative to A and B. Starting from these assumptions, he showed (mathematically) that there is no single consistent procedure for ordering alternatives for the group which will always satisfy the assumptions.

A number of other theoreticians in the area have challenged Arrow's theorem in various ways, particularly through challenging the "independence of irrelevant alternatives" assumption. The point here is that things are never evaluated in a vacuum but clearly are evaluated in the context of circumstance. A further charge is a pragmatic one: while Arrow proves inconsistencies *can* occur, in the great majority of cases likely to be encountered in the real world they would not occur, and if they did they probably would be of minor significance.

There are many other complicating factors in social choice, most of which have not been, and perhaps cannot be, dealt with in the systematic manner of Arrow's "impossibility theorem."² For example, there is the very fundamental question of whether the individual parties involved in a group choice exercise will communicate their true feelings and indicate their uncertainties, or whether they will falsify their feelings so as to gain the best advantage for themselves.

Further difficulties arise when we try to include in the treatment the effects of differences among the participants along the lines of intensity-of-feelings vs. apathy, or knowledge vs. ignorance, or "extended-sympathy" vs. selfishness, or partial vs. complete truthfulness; yet these are just the features of the social choice problem as we find it in practice.

To take as an ultimate goal the precise statement of social welfare in mathematical terms is, of course, nonsense. The differing experiences of individuals (and consequently differing assumptions) ensure that commensurability of values will never be complete. But this difficulty by no means relieves us of the obligation to seek value-commensurability and to see how far we can go in the quantitative assessment of utility. By making our values more explicit to one another we also make them more explicit to ourselves.

POTENTIAL CONTRIBUTIONS OF ELECTRONICS

Electronic media notwithstanding, none of the newer means of communication yet does what a direct face-to-face group meeting (town meeting, class bull session) does—that is, permit each participant to observe the feelings and gestures, the verbal expressions of approval

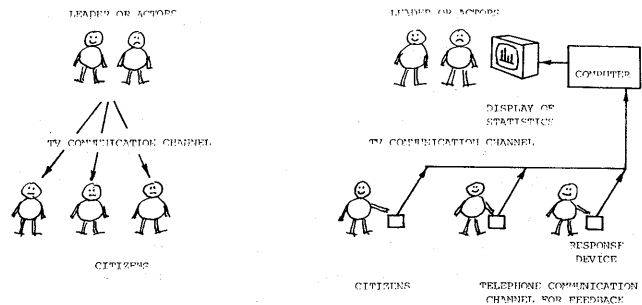


Figure 1—General paradigm for citizen feedback (right) added to top-down communication (left)

or disapproval, or the apathetic silence—which may accompany any proposal or statement. As a group meeting gets larger, observation of how others feel becomes more and more difficult; and no generally available technology helps much. Telephone conference calls, for example, while permitting a number of people to speak and be heard by all, are painfully awkward and slow and permit no observation of others' reaction to any given speaker. The new Picture-Phone will eventually permit the participants in a teleconference to see one another; but experiments with an automatic system which switches everyone's screen to the person who is talking reveals that this is precisely what is not wanted—teleconferees would like most to observe the facial expressions of the various conferees who are *not* talking!

One can imagine a computer-aided feedback-and-participation system taking a variety of forms all of which are more or less characterized by Figure 1. For example:

- (1) A radio talk show or a television "issue" program may wish to enhance its audience participation by listener or viewer votes, collected from each participant and fed to a computer. Voters may be in the studio with electronic voting boxes or at home where they render their vote by calling a special telephone number. The NET "Advocates" program has demonstrated both.
- (2) Public hearings or town meetings may wish to find out how the citizenry feel about proposed new legislation—who have intense feelings, who are apathetic, who are educated to the facts and who are ignorant—and correlate these responses with each other and with demographic data which participants may be asked to volunteer. Such a meeting could be held in the town assembly hall, with a simple pushbutton console wired to each seat.

- (3) Several P.T.A.s or alternatively several eighth grades in the town may wish to sponsor a feedback meeting on sex education, drugs, or some other subject where truthfulness is highly in order but anonymity may be desired. Classrooms at several different schools could be tied together by rented "dedicated" telephone lines for the duration of the session.
- (4) A committee chairman or manager or salesman wishes to present some propositions and poll his committee members, sales representatives or etc. who may be stationed at telephone consoles in widely separated locations, or may be seated before special intercom consoles in their own offices (which could operate entirely independently of the telephone system).
- (5) A group of technical experts might be called upon to render probability estimates about some scientific diagnosis or future event which is amenable to before-the-fact analysis. This process may be repeated, where with each repetition the distribution of estimates is revealed to all participants and possibly the participants may challenge one another. This process has been called the "Delphi Technique" after the oracle, and has been the subject of experiments by the Rand Corporation and the Institute for the Future,³ and by the University of Illinois.⁴ Their experience suggests that on successive interactions even experts tend to change their estimates on the basis of what others believe (and possible new evidence presented during the challenge period).
- (6) A duly elected representative in the local, state or national government could ask his constituency questions and receive their responses. This could be done through radio or television or alternatively could utilize a special van, equipped with a loudspeaker system, a rear-lighted projection/display device, and a number of chairs or benches which could be set up rapidly at street corners prewired with voter-response boxes and a small computer.

These examples point up one very important aspect of such citizen feedback or response-aggregation systems: that is that they can *educate and involve* the participants without the necessity that the responses formally determine a decision. Indeed the teaching-learning function may be the most important. It demands careful attention to how questions are posed and presented, what operations are performed by the computer on the aggregated votes and what operations are left out, how the results are displayed,

and what opportunity there is for further voting and recycling on the same and related questions.

Some skeptics feel that further technocratic invasion of participatory democracy should be prevented rather than facilitated—that the whole idea of the "computerized referendum" is anathema, and that the forces of repression will eventually gain control of any such system. They could be correct, for the system clearly presupposes competence and fairness in phrasing the questions and designing the alternative responses.

But my own fear is different. It is that, propelled by the increasing availability of glamorous technology and spurred on by hardware hucksters and panacea pushers, the community will be caught with its pilot experiments incomplete or never done.

THE STEPS IN A GROUP FEEDBACK SESSION

Seven formal steps are involved in a technologically aided interchange of views on a social-choice question:

- (1) The leader states the problem, specifies the question, and describes the response alternatives from which respondents are to choose.
- (2) The leader (or automated components of the system) explains what respondents must do in order to communicate their responses (including, perhaps, their degree of understanding of the question, strength of feeling, and subjective assessment of probabilities).
- (3) The respondents set into their voting boxes their coded responses to the questions.
- (4) The computer interrogates the voting boxes and aggregates the response data.
- (5) Preselected features of this response-aggregate are displayed to all parties.
- (6) The leader or respondents may request display of additional features of the response aggregate, or may volunteer corrections or additional information.
- (7) Based upon an a priori program, on previous results and/or on requests from respondents, the leader poses a new problem or question, re-starting the cycle from Step 1.

The first step is easily the most important—and also the most difficult. Clearly the participant must understand at the outset something of the background to any specific question he is asked, he must understand the question itself in nonambiguous terms, and he must understand the meaning of the answers or response alternatives he is offered. This step is essentially the

same as is faced by the designer of any multiple-choice test or poll, except that there is the possibility that a much richer language of response can be made available than is usually the case in machine-graded tests. Allowed responses may include not only the selection of an alternative answer, but also an indication of intensity of feeling, estimates of the relative probability or importance of some event in comparison with a standard, specification of numbers (e.g. allowable cost) over a large range, and simple expressions of approval ("yea!") or disapproval ("boo!").

The leader may have to explain certain subtleties of voting, such as whether participants will be assumed to be voting altruistically (what I think is best for everyone) or selfishly (what I think is best for me alone, me and my family, etc.). Further, he may wish respondents to play roles other than themselves (if you were a person under certain specified circumstances, how would you vote?).

He may also wish to correlate the answers with informedness. He may do this by requesting those who do not know the answer to some test question to refrain

from voting, or he can pose the knowledge test question before or after the issue question and let the computer make the correlation for him.

Insuring the participants "play fair," own up to their uncertainties, vote as they really feel, vote altruistically if asked, and so on, is extremely difficult. Some may always regard their participation in such social interaction as an advocacy game, where the purpose is to "win for their side."

The next two steps raise the question of what equipment the voter will have for communicating his responses. At the extreme of simplicity a single on-off switch generates a response code which is easily interpreted by the computer, but limiting to the user. At the other extreme, if responses were to consist of natural English sentences typed on a conventional teletypewriter—which would certainly allow great flexibility and variety in response—the computer would have no basis for aggregating and analyzing responses on a commensurate basis (other than such procedures as counting key words). Clearly something in between is called for; for example, a voting box might consist of ten on-off switches to use in various combinations, plus one to indicate "ready," plus one "intensity" knob.

An unresolved question concerns how complex a single question can be. If the question is too simple, the responses will not be worth collecting and will provide little useful feedback. If too complex, encoding the responses will be too difficult. The ten switches of the voting box suggested above would have the potential (considering all combinations of on and off) for $2^{10} = 1024$ alternatives but that is clearly too many for the useful answers to any one question.

It is probably a good idea, for most questions, to have some response categories to indicate "understand question but am undecided among alternatives" or "understand question and protest available alternatives" or simply "don't understand the question or procedures," three quite different responses. If a respondent is being pressured by a time constraint, which may be a practical necessity to keep the process functioning smoothly, he may want to be able to say, "I don't have time to reach a decision"; this could easily be indicated if he simply fails to set the "done" switch. Some arrangement for "I object to the questions and therefore won't answer" would also be useful as a guide to subsequent operations and may also subsume some of the above "don't understand" categories. Figure 2 indicates various categories of response for a six switch console.

The fourth step, in which the computer samples the voting boxes and stores the data, is straightforward as regards tallying the number of votes in each category and computing simple statistics. But extracting mean-

Identification of self (note: if one of 1, 2, 3 not switched assume unregistered or other party; if one of 4, 5, 6 assume other or none)	1) Republican 2) Democrat 3) Independent 4) Protestant 5) Catholic 6) Jew																																																																
Expressions of feeling and experience	1) Am intensely interested 2) Am mildly interested 3) Am uninterested 4) Daily experience 5) Occasional experience 6) No experience																																																																
Four numerical categories Two administrative categories	1) Less than 10% 2) 10 to 30% 3) 30 to 60% 4) Greater than 60% 5) Don't know 6) Don't understand																																																																
Three alternatives plus Three administrative categories	1) I want plan 1 2) I want plan 2 3) I want plan 3 4) Undecided as to plans 5) Object to available plans 6) Confused by procedure																																																																
Rank ordering of three alternatives, A, B, C	<table border="0"> <tr> <td>first choice</td> <td rowspan="3">}</td> <td>1) A</td> </tr> <tr> <td></td> <td>2) B</td> </tr> <tr> <td></td> <td>3) C</td> </tr> <tr> <td>second choice</td> <td rowspan="3">}</td> <td>4) A</td> </tr> <tr> <td></td> <td>5) B</td> </tr> <tr> <td></td> <td>6) C</td> </tr> </table>	first choice	}	1) A		2) B		3) C	second choice	}	4) A		5) B		6) C																																																		
first choice	}	1) A																																																															
		2) B																																																															
		3) C																																																															
second choice	}	4) A																																																															
		5) B																																																															
		6) C																																																															
Response to interpersonal communication of actors	<table border="0"> <tr> <td>as to</td> <td rowspan="6">}</td> <td>1) Miss Adams</td> </tr> <tr> <td></td> <td>2) Colonel Baker</td> </tr> <tr> <td></td> <td>3) Doctor Crank</td> </tr> <tr> <td>I</td> <td>4) Agree</td> </tr> <tr> <td></td> <td>5) Disagree</td> </tr> <tr> <td></td> <td>6) Am bored</td> </tr> </table>	as to	}	1) Miss Adams		2) Colonel Baker		3) Doctor Crank	I	4) Agree		5) Disagree		6) Am bored																																																			
as to	}	1) Miss Adams																																																															
		2) Colonel Baker																																																															
		3) Doctor Crank																																																															
I		4) Agree																																																															
		5) Disagree																																																															
		6) Am bored																																																															
To select one of 8 on each of two questions	<table border="0"> <tr> <td></td> <td></td> <td>A</td><td>B</td><td>C</td><td>D</td><td>E</td><td>F</td><td>G</td><td>H</td> </tr> <tr> <td>Question 1</td> <td rowspan="6">}</td> <td>1)</td><td>•</td><td>•</td><td>•</td><td>•</td><td>•</td><td>•</td><td>•</td> </tr> <tr> <td></td> <td>2)</td><td>•</td><td>•</td><td>•</td><td>•</td><td>•</td><td>•</td><td>•</td> </tr> <tr> <td></td> <td>3)</td><td>•</td><td>•</td><td>•</td><td>•</td><td>•</td><td>•</td><td>•</td> </tr> <tr> <td></td> <td>4)</td><td>•</td><td>•</td><td>•</td><td>•</td><td>•</td><td>•</td><td>•</td> </tr> <tr> <td>Question 2</td> <td rowspan="2">}</td> <td>5)</td><td>•</td><td>•</td><td>•</td><td>•</td><td>•</td><td>•</td> </tr> <tr> <td></td> <td>6)</td><td>•</td><td>•</td><td>•</td><td>•</td><td>•</td><td>•</td> </tr> </table>			A	B	C	D	E	F	G	H	Question 1	}	1)	•	•	•	•	•	•	•		2)	•	•	•	•	•	•	•		3)	•	•	•	•	•	•	•		4)	•	•	•	•	•	•	•	Question 2	}	5)	•	•	•	•	•	•		6)	•	•	•	•	•	•
		A	B	C	D	E	F	G	H																																																								
Question 1	}	1)	•	•	•	•	•	•	•																																																								
		2)	•	•	•	•	•	•	•																																																								
		3)	•	•	•	•	•	•	•																																																								
		4)	•	•	•	•	•	•	•																																																								
Question 2		}	5)	•	•	•	•	•	•																																																								
			6)	•	•	•	•	•	•																																																								
(dots under your answer indicate switches to be thrown)																																																																	

Figure 2—Sample categories of response for a six-switch console

ing from the data requires that someone should have laid down criteria for what is interesting; this might be done either prior to or during the session by a trained analyst.

It is at this point that certain perils of citizen feedback systems arise, for the analyst could (either unwittingly or deliberately) distort the interpretation of the voting data by the criteria he selects for computer analysis and display. Though there has been much research on voting behavior and on methods of analyzing voting statistics, instantaneous feedback and recycling poses many new research challenges.

That each man's vote is equally important on each question is a bit of lore that both political scientists and politicians have long since discounted—at least in the sense that voters naturally feel more intensely about some issues than about others. One would, therefore, like to permit voters to weight their votes according to the intensity of their feeling. Can fair means be provided?

There are at least two methods. One long-respected procedure in government is bargaining for votes—"I'll vote with you on this issue if you vote with me on that one." But in the citizen-feedback context, negotiating such bargains does not look easy. A second procedure would be to allocate to each voter, over a set of questions, a fixed number of influence points, say 100; he would indicate the number of points he wished the computer to assign to his vote on each question, until he had used up his quota of 100 points, after which the computer would not accept his vote. (Otherwise, were votes simply weighted by an unconstrained "intensity of feeling" knob, a voter would be rather likely to set the "intensity of feeling" to a maximum and leave it there.)

A variant on the latter is a procedure developed at the University of Arizona⁵ wherein a voter may assign his 100 points *either* among the ultimate choices *or among the other voters*. Provided each voter assigns some weight to at least one ultimate alternative an eventual alternative is selected, in some cases by a rather complex influence of trust and proxy.

Step five, the display of significant features of the voting data, poses interesting challenges concerning how to convey distributional or statistical ideas to an unsophisticated participant, quickly and unambiguously.

The sixth step provides an opportunity for non-planned feedback—informal exposition, challenges to the question, challenges to each other's votes, and verbal rebuttal—in other words a chance to break free of the formal constraints for a short time. This is a time when participants can seek to influence the future behavior of the leader—the questions he will ask, the

response alternatives he will include, and the way he manages the session.

EXPERIMENTS IN PROGRESS

Experiments to date have been designed to learn as much as possible as quickly as possible from "real" situations. Because the mode of group dialogue discussed above introduces so many new variables, it was believed not expedient to start with controlled laboratory experiments, though gradually we plan to make controlled comparisons on selected experimental conditions. But the initial emphasis has been on plunging into the "real world" and finding out "what works."

Experiments in a semi-laboratory setting within the university

In one set of experiments in the Man Machine Systems Laboratory at M.I.T. the group feedback system consists of fourteen hand-held consoles, each with ten on-off switches, a continuous "adjust" knob and a "done" switch. The consoles are connected by wire to a PDP-8 computer with a scope display output. Closed circuit television permits simulation of a meeting where questions are being posed and results aggregated at some distant point (e.g., a television station in another city) and where respondents may sit together in a single meeting room or may be located all at different places. Various aggregation display programs are available to the discussion leader, the simplest of which is a histogram display indicating how many people have thrown each switch. Other data reduction programs are also available, such as the one described above permitting voters to give a percentage of their votes to another voter. A variety of small group meetings, seminars and discussions have been held utilizing this equipment.

Two kinds of leadership roles have been tried. The first is where a single leader makes statements and poses questions. Here, among other things, we were concerned with whether respondents, if constrained to express themselves only in terms of the switches, can "stay with it" without too much frustration and can feel that they are part of a conversation. Thus far, for this type of meeting, we have learned the following:

- (1) Questions must be stated unambiguously. We learned to appreciate the subtle ways in which natural language feedback permits clarification of questions or propositions. Often the questioner doesn't understand an ambiguity in his statement—where a natural language response

from one or two persons chosen at random only for the purpose of clarifying the question is often well worth the time of others, though this by no means obviates the need to have some "I don't understand" or "I object" categories.

- (2) The leader should somehow respond to the responses of the voters. If he can predicate his next question or proposition on the audience response to the last one, so much the better. Otherwise he can simply show the audience that indeed he knows how the vote on the last question turned out and freely express his surprize or other reaction. In cases where the leader seemed as though he was not as interested in the response and simply ground through a programmed series of questions, the audience quickly lost interest.
- (3) Anonymity can be very important, and, if safeguarded, permits open "discussion" in areas which otherwise would be taboo. For example, we have conducted sessions on drug use, in which students, faculty, and some total strangers quite freely indicated how often they use certain drugs and where they get them. Such discussions, led unabashedly by students (who knew what and how to phrase the key questions!) resulted in a surprising freedom of response. (We made the rule that voters had to keep their eyes on the display, not on each others' boxes, though a small voting box can easily be held close to the chest to obstruct others' view of which switches are being thrown.) It was found especially important, for this kind of topic, not to display any results until all were in.

In the same semi-laboratory setting described above we have experimented with a second kind of leadership role. Here two or more people "discuss" or "act" and the audience continuously votes with "yea," "boo," "slow down and explain," "speed up and go on to another topic" type response alternatives. Voters were happy to play this less direct role but perhaps for a shorter time than in the direct response role described above. Again it proved of great importance that the central actors indicate that they saw and were interested in how the voters voted.

Experiments with citizen group meetings using portable equipment

As of this writing five group meetings have been conducted in the Massachusetts towns of Stoneham, Natick, Manchester, Malden and Lowell to assist the Massachusetts Department of Education in a program

of setting educational goals. In each case cross sections of interested citizens were brought together by invitation of persons in each community to "discuss educational goals." Four similar meetings were conducted with students and teachers at a high school in Newton, Massachusetts. (A similar meeting was also held in a church parlor in Newton to help the members of that church resolve an internal political crisis.) All groups ranged in size from twenty to forty, though at any one time only thirty-two could vote since but that number of voting boxes have been built.

The portable equipment used for these meetings, held variously in church assembly halls, school classrooms and television studios, features small hand-held voting boxes, each with six toggle switches, connected by wire to substations (eight boxes to a substation, each of the latter containing digital counting logic) which in turn are series connected in random order to central logic and display hardware. The display regularly used to count votes is a "nixie tube" type display of the six totals (number of persons activating each of the six switches). The meeting moderator, through a three position switch, can hold the numbers displayed at zero, set it in a free counting mode, or lock the count so that it cannot be altered. A second display, little used as yet, is a motorized bar graph to be used either to display histogram statistics or to provide a running indication of affective judgments such as agree with speaker, disagree, too fast, too slow, etc.

The typical format for these meetings was as follows. After a very brief introduction to the purpose of the meeting and the voting procedure itself several questions were asked to introduce members of the group to each other (beyond what is obvious from physical appearance) such as education, political affiliation, marital status, etc. An overhead projector has been used in most cases to ask the questions and record the answers and comments (on the gelatin transparency) since, unlike a blackboard, it need not be erased before making a permanent record. Following the introduction, the meetings proceeded through the questions, such as those illustrated by Figure 3, and those posed by the participants themselves. The categories of "object to question" or "other" were used frequently to solicit difficulties or concerns people had with the question itself—its ambiguity, whether it was fair, etc. Asking persons who voted in prepared categories to identify themselves and state, after the fact, why they voted as they did, was part of the standard procedure. Roughly twenty questions, with discussion, can be handled in 1½ hours.

After the meeting, evaluations by the participants themselves have suggested that the procedure does indeed serve to open up issues, to draw out those who

How are preschool children best prepared for school?

	(as school now exists)	(as school should be)
1) lots of parental love	9	11
2) early exposure to books	2	1
3) interaction with other kids	14	8
4) by having natural wonders and esthetic delights pointed out	1	5
5) unsure	4	3
6) object	0	1

Salient comments after vote ("as school now exists" and "as school should be" not part of question then): One man object to "pointed out" in 4), as it emphasized "instruction" rather than "learning." Discussion on this point. Someone else wanted to get at "encouraging curiosity." Another claimed, "That's what question says," and another "discover natural wonders." Consensus: "leave wording as is." Then a lady violently objected that the vote would be different depending on whether voter was thinking of school as it now existed or as it should be. Others agreed. Two categories added. Above is final vote.

Student attendance should be:

1) compulsory with firm excuse policy	11
2) compulsory with lenient excuse policy	0
3) voluntary, with students responsible for material missed	12
4) voluntary, with teachers providing all reasonable assistance to pupils who miss class	2
5) unsure	3
6) object	0

Comments centered on the feeling that some subjects require attendance more than others do. (Note the 0 vote on category 2) which is inappropriately self-contradictory.)

Figure 3—Typical questions and responses from the citizen meetings on educational goals

would otherwise not say much, and generally to provide an enjoyable experience—in some cases for three hours duration.

EXTENDING THE MEETING IN SPACE AND TIME

The employment of such feedback techniques in conjunction with television and radio media appears quite attractive, but there are some problems.

A major problem concerns the use of telephone networks for feedback. Unfortunately telephone switching systems, as they presently work, do not easily permit some of the functions one would like. For example, one would like a telephone central computer to be able to interrogate, in rapid sequence, a large number of memory buffers (shift registers) attached to individual telephones, using only enough time for a burst of ten or so tone combinations (like touchtone dial signalling), say about 1/2 second. Alternatively one might like to be able to call a certain number, and, in spite of a temporary busy signal, in a few seconds have the memory buffer interrogated and read over the telephone line. However, with a little investigation one finds that telephones were designed for random caller to called-

party connections, with a busy signal rejecting the calling party from any further consideration and providing no easily employed mechanism for retrieving that calling party once the line is freed.

For this reason, at least for the immediate future, it appears that for a large number (much more than 1,000) to be sampled on a single telephone line in less than 15 minutes, even for a simple count of busy signals, is not practical.

One tractable approach for the immediate future is to have groups of persons, 100 to 1,000, assembled at various locations watching television screens. Within each meeting room participants vote using hand-held consoles connected by wire to a computer, which itself communicates by telephone to the originating television studio. Figure 4 illustrates this scheme.

Ten or more groups scattered around a city or a nation can create something approaching a valid statistical sample, if statistical validity is important, and within themselves can represent characteristic citizen groups (e.g. Berkeley students, Detroit hardhats, Iowa farmers, etc.). Such an arrangement would easily permit recycling over the national network every few minutes and within any one local meeting room some further feedback and recycling could occur which is not shared with the national network.

Cable television, because of its much higher bandwidth, has the capability for rapid feedback from smaller groups or individuals from their individual homes. For example, even part of the 0-54 MHz band (considered as the best prospect for return signals⁶) is more than adequate theoretically for all the cable subscribers in a large community, especially in view of time sharing possibilities.

The above considerations are for extensions in space.

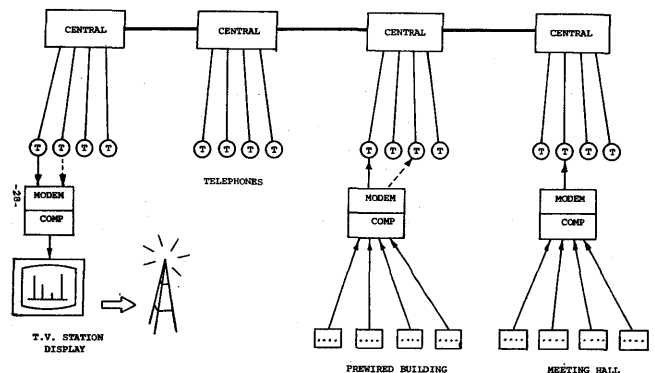


Figure 4—Multi-group arrangement for television audience response

One may also consider extensions in time, where a single "program" extends over hours or days and where each problem or question, once presented on television, may wait until slow telephone feedback or even mail returns of an IBM card or newspaper "issue ballot," variety come in.

Development of such systems, fraught with at least as many psychological, sociological, political and ethical problems as technological ones, will surely have to evolve on the basis of varied experiments and hard experience.

REFERENCES

- 1 J VON NEUMANN O MORGANSTERN
Theory of games and economic behavior
Princeton University Press 2nd edition 1947
- 2 K ARROW
Social choice and individual values
John Wiley New York 1951
- 3 N DALKEY O HELMER
An experimental application of the Delphi Method to the use of experts
Management Science No 9 1963
- 4 C E OSGOOD S UMPLEBY
A computer-based system for exploration of possible systems for Mankind 2000
Mankind 2000 pp 346-359 Allen and Unwin London
- 5 W J MACKINNON M K MACKINNON
The decisional design and cyclic cooperation of SPAN
Behavioral Science Vol 14 No 3 pp 244-247 May 1969
- 6 *The third wire: cable communication enters the city*
Report by Foundation 70 Newton Massachusetts March 1971
- 7 C H STEVENS
Citizen feedback, the need and the response
MIT Technology Review pp 39-45 Cambridge Massachusetts

Structuring information for a computer-based communications medium*

by STUART UMPLEBY

University of Illinois
Urbana, Illinois

Several years ago Prof. Charles E. Osgood suggested that it might be possible to develop a program for a computer-based education system which would eventually allow the public, possibly at a world's fair, to "explore the future."¹ Such an "exploration" would be useful both for education and for social science research. This paper is a progress report on the continuing development of that "exploration of alternative futures" using the PLATO system (see Figure 1).

The educational function of the exploration is accomplished by exposing the "explorer" to four types of information:

1. A list of developments possible in the future.
2. The model of "reality interaction" used in the computer program.
3. The decision-making procedure, including making investments to change probabilities.
4. The operation of a teaching computer.

The data from these exercises can reveal which developments people consider desirable, which developments they are most familiar with, and which developments they are most interested in. Data from several demonstrations of early versions of the exploration will be discussed later in this paper.

In short this work originated neither as an attempt to develop software for a new communications medium nor from an interest in conducting on-line Delphi

studies as a decision-making aid. The direction in which the research moved resulted in part from the nature of the medium and in part from our concern with increasing public participation in decision-making processes.

THE EVOLUTION OF THE PROJECT

After the project had been under way for a year or two, it appeared that this work could have applications beyond simply education and social science research. This belief resulted from the projected growth of the PLATO system and the ease with which the computer program for the exploration could be modified to deal with problems other than the general future of mankind.

A glance at the past

The PLATO III system, which we have been using, is capable of operating 20 terminals simultaneously. However, the PLATO IV system, scheduled for completion in 1974 or 1975, is being designed to operate 4000 terminals simultaneously (see Figure 2).

When the first wave of interest in computer-aided instruction began in the early 1960s there were basically two questions which had to be answered. First, could students learn educational material as rapidly and retain it as well if they were taught using a computer terminal rather than by sitting in a classroom? Second, was computer-aided instruction economically competitive with classroom instruction? After a decade of educational experiments at the University of Illinois and elsewhere, the answer to the first question is emphatically yes.² But computer-aided instruction is not economically competitive using the technology available in 1960. It was this realization—that the crucial problem lay in the cost of the equipment and its operation—that led to the invention at the University of Illinois of

* The research described here was conducted using the PLATO system at the Computer-based Education Research Laboratory of the University of Illinois at Urbana-Champaign. The laboratory is supported in part by the National Science Foundation under grants NSF GJ81 and GJ 974; in part by the Advanced Research Projects Agency under grant ONR Nonr 3985 (08); in part by Project Grant NPG-188 under the Nurse Training Act of 1964, Division of Nursing, Public Health Service, U.S. Dept. of Health, Education and Welfare; and in part by the State of Illinois.



Figure 1—The PLATO III system provides each student with an electronic keyset as a means of communicating with the computer and a television display for viewing information selected or generated by the computer

the plasma display panel and the development of the PLATO IV system.³

A computer-based education system which is both educationally effective and economically competitive will in all probability be adopted throughout the United States and around the world in due course. With the prospect of teaching computer terminals in most classrooms, if not most homes, within a few decades, we began to think of this equipment as a new kind of mass communications medium.

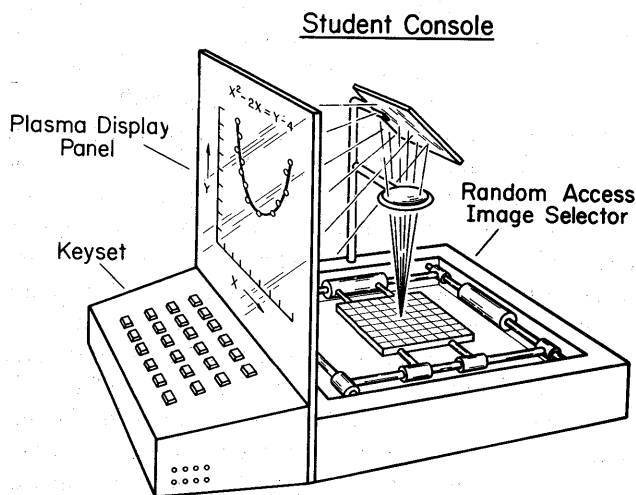


Figure 2—Using terminals such as the one pictured above, the PLATO IV system, scheduled for completion in 1974, will provide a high quality color display at low cost. The terminals will be connected to the computer over standard voice-grade telephone lines

Generations of communications media

If radio and television are thought of as first and second generation mass communications systems, then perhaps the PLATO system could be thought of as a forerunner of a third generation mass communications system. Originally, we felt that built-in feedback would be the characteristic distinguishing computer-based communications systems from radio and television.

However, due to the feedback possibilities of cable television, it now seems more appropriate to list *four* "generations" of electronic communications media now existing in at least prototype form.

1. *Radio* transmits audio messages from the center to the periphery.
2. *Television* transmits audio and visual messages from the center to the periphery.
3. *Cable television* provides a great increase in the number of available channels and the possibility of both passive feedback (monitoring what people watch) and active feedback (for example, voting by pressing a button on the television set).
4. *Computer-based communications systems* have several new characteristics.
 - a. *Less simultaneity*: Although many people may be using the program, each may be in a different part of the program. Thus everyone on one "channel" does not see the same thing at the same time.
 - b. *Less evanescence*: With radio and television a listener or viewer cannot go back if he misses a word or sentence (unless he has a tape recorder). With PLATO each individual progresses at his own rate. The display does not change until he wants it to, and he can go back to review previous displays.
 - c. *Viewer-designed programs*: With PLATO the viewer can ask for additional information or can jump ahead if he becomes bored, thus to some extent designing his own program.

For the sake of clarity it should be noted that the displays for computer-based communications systems are generally static, like color slides, rather than dynamic, like movies or television. Messages are conveyed primarily by the use of words, frequently supplemented by tables or graphs and occasionally by drawings and pictures. However, the PLATO IV equipment, as contrasted with the PLATO III equipment, will use audio as well as visual messages and will be better suited to simple moving displays such as dancing stick figures or plotting out a graph.

Applying the exploration to specific issues

Even before we began thinking of the PLATO system as a new kind of mass communications medium and not simply as a teaching device, we had thought of writing explorations on a variety of different topics such as disarmament, the future of education, and urban planning. But with the probable widespread acceptance of computer-based education in the next few decades, it seemed that our work suggested the possibility of using this equipment as a medium between planners and the public for exchanging information and opinions regarding community goals.⁴

The adaptability of the exploration resulted from the fact that the decision-making framework could remain the same for any problem area and that only the information units with the matrix giving the relationships between them would need to be changed.

Thus the projected expansion of the PLATO system and the ease with which the exploration could be modified to deal with specific issues resulted in our thinking of the teaching computer as a new kind of mass communications medium particularly suited to discussions among different interest groups about the long range goals of a community. But how does one present information on this new medium?

HOW DOES ONE DESCRIBE THE FUTURE?

Programming an exploration of the future required suggesting possible future developments in a way which emphasized their probabilistic nature and in a way which could be easily manipulated by the explorer. We needed a method of presenting possible future developments so that people could request additional information, make "investments" which would alter the initial probabilities, and see the possible secondary effects of their actions. Consequently we assumed that the future, and also the present and the past, can be described using "information units."

Features of information units

The features or components of an "information unit," as described in an earlier report, were (1) a short descriptive statement, (2) a background paragraph, and (3) an associated probability or other measure.⁵ It now seems necessary to expand and revise this list of features.

The *background information* (2) can involve graphs, charts, pictures, drawings, and tape recordings in addition to written information.

Complete *measurement* (3) of an historical occurrence

requires the measurement itself, the date the measurement was made, and some indication of measurement error. The measurement of a forecast requires the forecasted value of a parameter, the date at which that value of the parameter is expected to occur, some indication of certainty about the forecast, and also the date at which the forecast is made.

With respect to graphs of information units, the horizontal axis will always be time. The vertical axis which we have used so far for developments has been probability, ranging from 0 to 100 percent. This kind of scale requires a specific, identifiable event such as 50 percent of the nation's schools having computer-based education equipment or 50 percent of the population favoring the legalization of marihuana. A superior method of forecasting would lend itself more easily to validation and would make possible the measurement of progress toward a goal as the years pass by. Such a scale is suggested by the previous format. Rather than measuring the probability of a particular level of distribution of a technology, one can simply measure the distribution itself. Similarly one can estimate the percentage of the population which will favor a social development rather than the probability of a particular degree of acceptance. Regarding the development of a technological capability, as opposed to its diffusion once it is developed, one could list the stages of development ranging from the original concept through experimentation and prototype construction to the first production model. However, this kind of measure would use an ordinal rather than an interval scale.

An important new feature which should be noted explicitly when developing lists of information units is *the group or person suggesting a particular idea as important* (4) and worthy of attention. This information can usually be deduced either from the name and affiliation of the person writing the paper or from the list of people who took part in a Delphi exercise. The person or group originating an idea is frequently recorded. However, the thought behind recording this data is usually either to aid in locating additional background material or to give credit where it is due. But such information is also politically relevant. It is needed so that other forecasters, public officials, and especially the general public, will know whose ideas about the future of society are represented in the total set of forecasts and social indicators now being generated. People in different walks of life, in different socio-economic groups, will be subjected to different stresses in their daily lives. Consequently they will define different "problems" as being important for mankind to solve. The intervention of politics into forecasting cannot be avoided, we can only try to be aware of

possible sources of bias so that the interests of all groups will be as fairly represented as possible.⁶





Categories of information units

Information units can be divided into four categories:

1. *Developments*, including both social and technological developments, refer to new characteristics of the social system.

2. *Initiatives* are actions taken by a group or an individual.
3. *Events* are sudden or unanticipated occurrences.
4. System variables, now more commonly called *social indicators*, are measures of a system which fluctuate in time.

Two criteria are used to distinguish among these four kinds of information units: the shape of the graph over time and the extent of human control.

Type of information unit	Shape of graph	Human control
Development	S-curve 	Many small decisions
Initiative	Step function 	Single large decision
Event	Spike function 	Very little control
Social indicator	Fluctuation 	Regular adjustments

The earlier report suggested that the four categories of information units could be thought of either as "change-producing factors" (developments, events, and initiatives) or as "system variables" (social indicators). The distinction between these two larger categories lies in the period of time during which the information unit can usefully be of interest. System variables or social indicators are of interest over an indefinite period of time and so are used to monitor the behavior of social systems. Change-producing factors can occur in a period of hours or years but are of little interest outside of the period of time during which they are producing change in the social system.

Most mathematical models deal with the relationships among several "system variables," such as population, per capita income, gross national product, and capital investment in agriculture. Delphi studies usually concern themselves only with "change producing factors." An ideal exploration of the future would use both system variables and change-producing factors.

A new literary form

Every new communications medium seems to generate its own distinctive forms for structuring information. The printing press made possible newspapers, journals, and novels. Films greatly extended the use of animated cartoons and led to zoom and pan shots,

parallel editing and special visual effects. Radio and television produced the talk show, 15 minute news, commercials, and spot announcements. The mimeograph machine was best suited to the leaflet, the working paper, and the "underground press." The Xerox machine promoted letter writing to multiple recipients and extended the readership of journal and magazine articles. It is not surprising that computer-based communications media also seem to be developing their own literary form.

Branching sequences and mathematical algorithms, so useful in "individualized instruction," create a demand for literature in which statements and paragraphs can be rearranged, dropped or added. Scripts or programs which follow the single logical sequence of the essay are criticized by the managers of the medium as "not taking full advantage of the capabilities of this kind of system." In such cases there is pressure to either rewrite the material or present it using a different communications medium.

Rather than an articulate text with an interesting introduction and a good summarizing conclusion, the material written for a computer-based communications medium, particularly when it deals with public issues, emphasizes alternatives and their consequences and concisely stated, measurable events. With this medium a person can describe his ideal future without having to give a speech or write an essay or book. Furthermore, his views can be easily compared

or combined with the "ideal futures" of other people, thereby informing the explorer, the programmer, and the general public what visions are dancing in the heads of their fellow citizens.

We have long needed a literary style which, rather than imposing a particular idea, tends to draw out new ideas, and which tends simply by its form to make normally implicit assumptions explicit so that they can be challenged. When people see that they disagree about the relationships between developments or events they may discover that their disagreements are not about basic values or goals as much as they are about factual questions such as what does in fact lead to what.

The future-oriented literary form of the Delphi Method is different in several important respects from the present and past-oriented literary forms more commonly used today.⁷ The essay form, whether a newspaper report, a magazine article or a book, is most useful for developing a single idea to a certain degree of detail. A story related in this way has only one plot and all the subplots are related in the same way for all readers. Thus for the reader it is not a very personalized artistic form, no matter how weird one's powers of interpretation. It is little wonder that one criterion of quality in a short story or novel has been the range of interpretations or meanings which can be drawn out of it. This practice might be thought of as bestowing cudos for the ability to transcend the limitations of the medium. Imagine the artistry possible if a literary form could be designed which had nearly all of the strong points of the essay but reduced or eliminated some of the limitations!

The essay requires only the passive involvement of the reader. Fantasy and relationships with previous knowledge or experience can be brought into play, but the new ideas which are generated cannot be tested out in the story itself. Literary essays, reports, stories, even films, plays, and melodramas are closed ended and are characterized by high certainty. Events do or do not happen. The closest thing to the hypothetical or probabilistic is the scientific report with its margins of error and the assumption that refutation is possible. But the scientific report states facts, not possibilities. Even science fiction stories while beginning from a hypothetical situation follow a fixed course to a unique conclusion.

Robert Theobald's *Teg's 1994* a mimeographed, alterable account of a small girl's possible future world is one example of rumblings of a demand for new literary forms which are flexible, probabilistic, open-ended and user-controlled, thereby permitting active involvement of the reader.⁸ A more conditional and manipulatable style of literature will not be very satisfying and may

be downright disconcerting to some people. It will probably be most satisfying for people who have a high tolerance for uncertainty and ambiguity and who appreciate being asked for their judgments as well as being given someone else's. A plot not subject to influence other than interpretation is suitable for a past not subject to influence other than interpretation. A future susceptible to action and open to invention requires a medium which invites action and encourages invention.

THE EXPLORATION AND SOME RESULTS

With the preceding background on how our thinking about the project has evolved and the refinement of what is meant by information units, we shall now pause in our speculations for a look at the present version of the exploration and the data which has been collected so far.

An outline of the exploration

The decision-making procedures in one cycle of the 40 information unit exploration were as follows:

1. From a list of the 35 social and technological developments programmed into the computer the explorer chooses a development whose probability he would like to change.⁹ The object is to make more probable those developments which the explorer considers desirable and less probable those developments which the explorer considers undesirable. However, desirable developments may have undesirable secondary effects, and undesirable developments may have some desirable secondary effects.
2. The explorer makes an "investment" (an indication of desirability) between -100 and $+100$, where -100 would mean that the development is maximally undesirable, 0 would mean that the development is neither desirable nor undesirable, and $+100$ would mean that the development is maximally desirable. An investment such as $+50$ would mean that the development is moderately desirable. In the present version, no limit is placed on the total amount which can be invested in an exploration. 100 units could be invested during each cycle.
3. The computer shows in table form the secondary effects of the explorer's immediately preceding investment according to the estimates of secondary effects put into the computer by the programmer. For each development listed as a

TABLE I—Demonstrations with Recorded Data

Date	Number of people	Group	Notes
1. 3/9/68	?	?	1. 15 information units 2. uses GENERAL language
2. 3/13/68	?	?	3. only comment data (Always same as preceding demonstration except as noted)
3. 5/12/68	17	Social Science Undergraduates	1. uses TUTOR language 2. no comment mode 3. primary development (PD) selected randomly and not recorded in data 4. shows background paragraph for PD 5. investment is only +, 0, - 6. asks for relationship of PD to 4 predetermined secondary developments, relationship must be given as +, 0, - 7. select 5 to follow, option to see 3 other secondary effects 8. 4 stage oracle 9. main calculation sequence is part of special version of TUTOR 10. secondary effects matrix read in by paper tape (+, 0, -)
4. 7/10/68	10	Social Science Faculty & Graduate Students	1. magnitude on investment can go to ± 99 2. asks for numbers of 3 developments which might affect PD and how they will affect PD (+, -)
5. 10/9/68	11	Undergraduates from several disciplines	1. indirect investment possible in up to 5 developments 2. no question on what developments affect PD 3. magnitude of investment can go to total of 100 in one cycle
6. 2/1/69	6	local press	
7. 2/17/69	7	Undergraduates from several disciplines	
8. 3/3/69	15	Political Science Graduate Students	1. Built-in comment mode
9. 3/4/69	9	Social Science Undergraduates	
10. 3/17/69	7	Political Science Graduate Students	
11. 5/12/69	13	Education Professors	1. data printout at end of each cycle, gives PD, cycle number, and probabilities for all 15 developments 2. calculation sequence and secondary effects matrix built into Delphi program
12. 2/14/71	9	Landscape Architecture Graduate Students	1. 40 information units, 35 developments and 5 events 2. PD selected by explorer 3. background paragraph for PD not automatically displayed, must be requested 4. no indirect investment 5. does not ask for estimates of 4 secondary effects relationships 6. automatic selection of secondary effects, only those whose probabilities are changed by the investment in PD 7. secondary effects matrix with relationships up to ± 3
13. 2/20/71	5	WBBM reporter and friends	
14. 2/27/71	8	Urban planning Graduate Students	
15. 2/28/71	13	Graduate Students in secondary education	
16. 3/6/71	6	Graduate Students in religion	

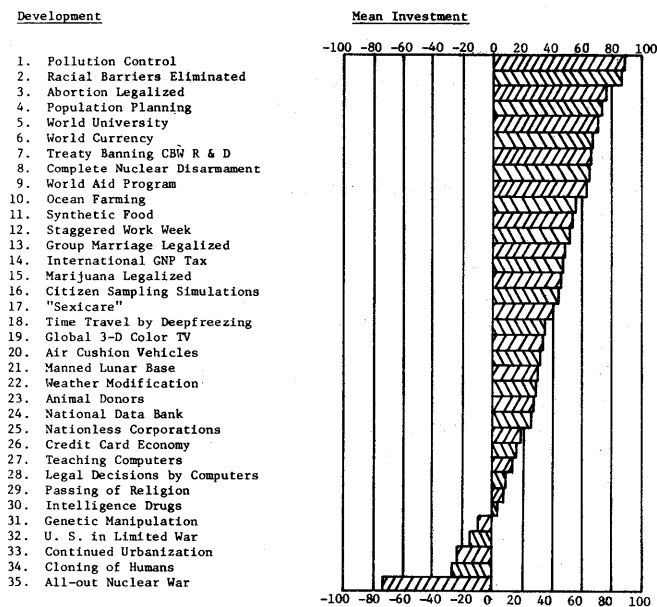
secondary effect the computer displays the old probability (before the investment) and the new probability (after the investment) and the change in probability (the difference between the two).**

4. An oracle message is displayed. Oracle is a verbal message telling which developments are likely to happen in the year 2000 and which are not likely to happen, on the basis of the current probability of each development in the exploration.
5. At the end of each cycle the computer performs several random calculations to determine whether an "event" occurs. If an event does occur, a background paragraph about it is presented and then its effects on the probabilities of the social and technological developments are shown by a table of secondary effects.

Tradeoffs due to hardware and software limitations

The primary limitation on the complexity of these explorations has been the amount of computer memory allocated to what is called student bank, the number

TABLE II—Rank Order of Developments by Mean Investment (Data from demonstrations 12–16)



** For a discussion of the mathematical model used in the exploration and the decisions which have to be made by the programmer, see Reference 5.

TABLE III—Rank Order by Frequency with which the Development was Selected for Consideration*

Development	Number of times Chosen for Investment
1. Pollution Control	31
2. Racial Barriers Eliminated	25
3. Complete Nuclear Disarmament	23
4. Abortion Legalized	21
5. All-out Nuclear War	19
6. Population Planning	18
7. Continued Urbanization	15
8. Genetic Manipulation	15
9. Citizen Sampling Simulations	13
10. Cloning of Humans	13
11. Group Marriage Legalized	13
12. International GNP Tax	13
13. Treaty Banning CBW R & D	12
14. World Aid Program	12
15. World University	12
16. Marijuana Legalized	11
17. Passing of Religion	11
18. U. S. in Limited War	11
19. Air Cushion Vehicles	10
20. Credit Card Economy	10
21. Legal Decisions by Computers	10
22. Manned Lunar Base	10
23. "Sexicare"	10
24. Intelligence Drugs	9
25. National Data Bank	9
26. Animal Donors	8
27. Ocean Farming	8
28. Synthetic Food	8
29. Global 3-D Color TV	7
30. Staggered Work Week	7
31. Weather Modification	7
32. Nationless Corporations	6
33. Teaching Computers	6
34. World Currency	4
35. Time Travel by Deepfreezing	3

* Data from demonstrations 12–16.

of words accessed by only one terminal. The number of variables can be increased somewhat by packing, but of course this procedure involves a limit as well.

The 15 information unit explorations, particularly the later ones, involved a larger number of decision-making operations in each cycle, such as indirect investment (what other developments are likely to affect the occurrence of the development under consideration) and asking for estimates by the explorer of some of the probable secondary effects of the development being considered.

When the number of information units was expanded to 40, the number of decision-making operations performed by an explorer in each cycle was decreased. This reduction resulted both from the demand for more

TABLE IV—Rank Order by Frequency with which Background Information was Requested*

Development	Number of times background information was requested
1. Cloning of Humans	22
2. "Sexicare"	21
3. Citizen Sampling Simulations	17
4. Pollution Control	15
5. Racial Barriers Eliminated	15
6. Treaty Banning CBW R & D	15
7. Complete Nuclear Disarmament	14
8. Group Marriage Legalized	14
9. Passing of Religion	14
10. Population Planning	14
11. Abortion Legalized	13
12. Animal Donors	13
13. Genetic Manipulation	11
14. All-out Nuclear War	10
15. Air Cushion Vehicles	9
16. Continued Urbanization	9
17. International GNP Tax	9
18. Ocean Farming	8
19. World University	8
20. Intelligence Drugs	7
21. Legal Decisions by Computer	7
22. National Data Bank	7
23. Staggered Work Week	7
24. Nationless Corporations	6
25. U. S. in Limited War	6
26. World Aid Program	6
27. Marijuana Legalized	5
28. Synthetic Food	5
29. Credit Card Economy	4
30. Manned Lunar Base	3
31. Teaching Computers	3
32. Time Travel by Deepfreezing	3
33. Global 3-D Color TV	2
34. World Currency	2
35. Weather Modification	1

* Data from demonstrations 12-16.

variables caused by more information units and from the addition of some more sophisticated computer operations.

Discussion of the data

Since work on developing the computer program began in the fall of 1966, sixteen demonstrations of an exploration of the future have been given during which data was recorded. Numerous demonstrations were given during which data was not collected. Table I lists by dates the demonstrations during which data was collected. The number of people participating in the demonstration and the background of the group are given in the second and third column. The right-

hand column contains notes about the nature of the program, such as the number and categories of information units and the decision-making operations which were added or dropped since the previous demonstration.

The first 11 demonstrations were of an exploration having only 15 information units, all of which were either social or technological developments. Demonstrations 12-16 were of an exploration having 40 information units—35 social and technological developments and 5 events. Table II lists the 35 social and technological developments according to the mean investment in each development.

Table III lists the developments in order according to the number of times each development was chosen as an object of investment. The greater the number of people who choose to invest in a development, the less influence each person has in determining its mean investment.

TABLE V—Answers to Questionnaire at End of Exploration*

1. Is the outcome close to or far away from the future you had hoped to achieve?	
a. very close	2
b. close	14
c. slightly close	19
d. slightly far	2
e. far	1
f. very far	1
2. If you had it to do all over again, would you change any of your investments?	
a. yes	25
b. no	15
3. I found the information in the background paragraphs	
a. helpful	33
b. not helpful	4
c. wrong	2
4. I found the instructions on what to do next	
a. sufficient	34
b. insufficient	2
c. repetitious	1
d. badly written	2
5. All in all I found the Delphi exploration to be	
a. loads of fun	10
b. fun	25
c. a bore	0
d. a complete waste of time	1
6. Sex	
a. F	12
b. M	24
7. Year in school	
a. freshman	2
b. sophomore	2
c. junior	0
d. senior	4
e. graduate student	20
f. professor	15

* Data from demonstrations 12-16.

The number of times that background information was requested for each development is shown in Table IV. There are no doubt a variety of reasons for requesting background information. The more prominent items in the list seem to be those with which people are least familiar. Background paragraphs are also frequently requested on subjects which the explorer is familiar with, probably in order to test the expertise of the people writing the program. Similarly, controversial subjects seem to be called up in order to divine the political opinions of the programmer.

Ten minutes before the end of the hour each person is asked to type a code word in order to jump to a series of questions at the end of the exploration. Seven of these questions are listed in Table V. Question 2 asks, "If you had it to do all over again, would you

change any of your investments?" If the explorer answers, "yes," he is then asked to list the numbers of the developments in which he would make a different investment. Table VI lists the 35 developments in order from the most to the least frequently mentioned in responses to this question.

Not all of the people who worked through at least part of the exploration completed the questionnaire at the end of the exploration. Fifty-four people participated in 6 demonstrations. Of that 54, 39 completed the questionnaire.

A non-random sample of people

It should be stressed that the people who took part in these demonstrations were not randomly selected from the population at large. They were not even randomly selected from the university community. The disciplines represented are suggested by the groups listed in Table I. The data in Table V, questions 6 and 7, shows the distribution of people according to sex and year in school. An open-ended question on political viewpoints stimulated frequently extended critiques of American society. My own interpretation of their answers indicates that there were two radical liberals, 10 liberals, and 2 people between middle and right wing.

Furthermore, the people were not randomly selected in terms of their interest in the exploration. With the exception of a few students in political science classes, all of the explorers to date have asked to work through the exploration or have responded to the encouragement of a friend to do so. The most frequent pattern is for an interested faculty member to bring along either a group of faculty members or a class of graduate students. We have not yet systematically sought representative samples since we are still primarily concerned with the development of a more interesting program from the viewpoints of both education and research. Consequently this data is presented only as a very preliminary indication of the kinds of responses that can be obtained when using a computer-based communications medium to discuss an area of public policy. The responses should *not* be interpreted as representative of how the American people or even university people would rate the desirability of the developments listed. The data *is* useful in indicating how many possible future developments can be considered by an educated person in a given period of time.

Measures of performance

Despite extensive instructions at the beginning of the exploration, a few people have great difficulty figuring

TABLE VI—Developments in which People Would have Changed Their Investment*

Development	Number of times listed
1. Continued Urbanization	5
2. Pollution Control	4
3. U. S. in Limited War	4
4. Complete Nuclear Disarmament	3
5. Legal Decisions by Computers	3
6. Ocean Farming	3
7. Passing of Religion	3
8. Population Planning	3
9. Racial Barriers Eliminated	3
10. Treaty Banning CBW R & D	3
11. World Aid Program	3
12. Cloning of Humans	2
13. International GNP Tax	2
14. National Data Bank	2
15. Synthetic Food	2
16. World Currency	2
17. Abortion Legalized	1
18. All-out Nuclear War	1
19. Genetic Manipulation	1
20. Group Marriage Legalized	1
21. Intelligence Drugs	1
22. Nationless Corporations	1
23. "Sexicare"	1
24. Staggered Work Week	1
25. Teaching Computers	1
26. Weather Modification	1
27. Air Cushion Vehicles	0
28. Animal Donors	0
29. Citizen Sampling Simulations	0
30. Credit Card Economy	0
31. Global 3-D Color TV	0
32. Manned Lunar Base	0
33. Marijuana Legalized	0
34. Time Travel by Deepfreezing	0
35. World University	0

* Data from demonstrations 12-16.

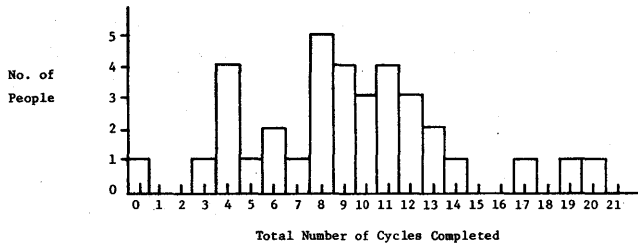


Figure 3—Total number of cycles completed (Data from demonstrations 12, 14, 15, 16)

out what they are supposed to do. This is shown by the fact that in Figure 3 several people were able to complete only a very few cycles.

The time allotted for the exploration was in most cases one hour. The demonstration on 2/20/71 extended to about an hour and twenty minutes. For the five demonstrations of the 40 information unit game the mean number of cycles completed was 10.3. For the four demonstrations in which only one hour was available, a mean of 8.9 cycles were completed with the mode being 8 and the median 10.

The number of people requesting a particular number of background paragraphs is shown in Figure 4. The mean number of background paragraphs requested was 8.8. The mode was 6 and the median 11.5.

The number of people having a particular number of random events occur in their exploration is given in Figure 5. The mean number of events in an exploration was 1.7. The mode was 1 and the median 2.5.

Figure 6 shows the number of people who made a certain number of comments. The mean number of comments was 1.6. The mode was 1 and the median 2. Everyone was explicitly asked for comments, suggestions, and criticisms as one part of the questionnaire at the end of the exploration. Consequently there were very few people who made no comments, or a response such as "none" in answer to that question. Those

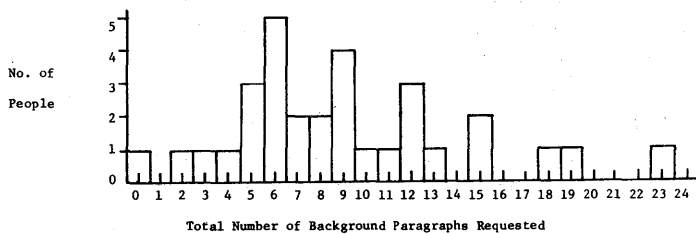


Figure 4—Total number of background paragraphs requested (Data from demonstrations 12-16)

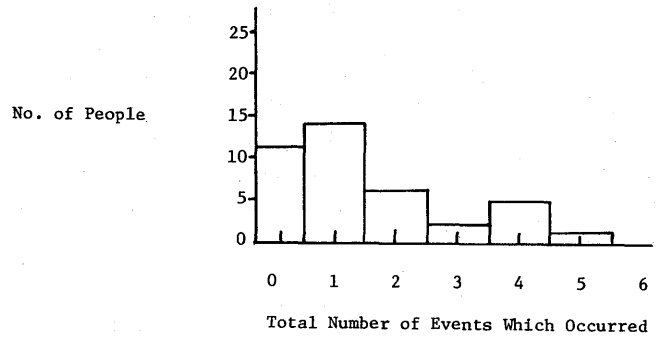


Figure 5—Total number of events which occurred (Data from demonstrations 12-16)

people who made only one comment did so in reply to the specific request and did not interrupt the exploration in order to go into the "comment mode."

Comments by explorers

The comments made by the participants during the demonstrations reflected a variety of criticisms, suggestions, questions, and general reactions. These can be grouped in the following categories:

1. *Technical errors.* Debugging is an activity familiar to all computer programmers. Debugging a program on a teaching computer involves calling in some friends who either have a knack for making things go wrong or who find a malicious glee in outwitting a computer. Examples of technical errors would be that the computer does not accept a negative number when it should, or it accepts a letter when it should accept only numbers. A few participants were quick to point out such errors: "Ha! You made a mistake."

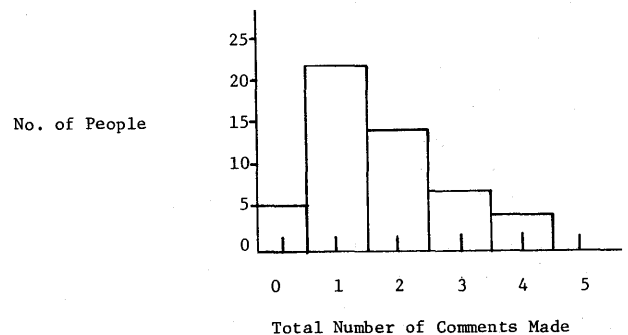


Figure 6—Total number of comments made (Data from demonstrations 11-16)

2. *Instructions.* Nearly every exploration produces suggestions for clarifying the instructions. For example, "I did not understand how to invest and make a meaningful contribution toward my goal." "It would be helpful if we were given a reminder to enter our choices and outcomes on the data sheet." "Need more explanation of what you mean by investment."

3. *The purpose of the exploration.* In a game in which no score is kept, where the object is to make the desirable probable and the undesirable improbable, expressions of confusion are to be expected. "The final results of the game or the goals were not clear to me." Some people seem to have difficulty conceptualizing complex systems. "I had difficulty realizing the issues. Consequently, I'm not sure I was making intelligent decisions." On the other hand, some people experience "an awakening of the sense of the future of man."

4. *Important items not included.* Some of the most thoughtful and helpful comments deal with what is not included in the exploration. "I did not think that there were enough policy factors involved in the issues contained here. For example I would have liked to see the question of manipulation of the individual considered more explicitly—in other words questions about attitudinal changes." "Complete disarmament rather than nuclear disarmament should be one of the futures included."

5. *The original probabilities.* Explorers sometimes question the actual estimates. "All out nuclear is not possible." And sometimes they question who made the estimates. "I am curious as to how the original probabilities for the given issues were determined."

6. *Background paragraphs.* The brief background information was challenged usually for not giving sufficient attention to consequences, regulation, or alternative solutions. "Nothing was mentioned about the effects of pressure on the surface the air cushion vehicle is going to be passing over. Until the ecological effects of such an apparatus over a natural surface are determined, I would seriously question such an apparatus." "In such items as cloning of humans, crucial matters would be the regulations that go along with the process. It is difficult to decide if it is good or bad without at the same time fixing some of the regulations." "Legislation or executive action is not needed to stabilize population growth rate—education is the answer to this problem."

7. *Secondary effects.* Questions about secondary effects ranged from who determined them and the logic used, to amazement that there were so many. For example, "How does a 100 percent investment in world aid program reduce probabilities of ocean farming and synthetic food?" "I am unclear as to how the associ-

ations were decided upon after I had made my investment or an event had happened. Some of them did not make sense to me. Like some of the considerations were left out." "I do not see what happened at this point which caused the probability of limited war to increase." "I would have been interested in how various choices affected other events specifically, e.g., a little more about why one probability caused a particular change in another variable." "Frightening how one decision influences so many others you never took into consideration when making your initial decision. On some do not see off hand how the influence came about."

8. *What makes the exploration enjoyable.* "I like when you add the event to make it more exciting." "I was angry at being told what to do in such a demanding way by a machine. My hands felt slapped each time I made a mistake." "I would have liked to have finished in order to see the full picture of the world. Occasionally became bored, however, perhaps due to each step being handled in the exact same manner as the previous."

9. *The dangers or opportunities implied by this technology.* Only a very few people remarked on the potential of this kind of equipment and this kind of program for radically altering the process of citizen participation in planning. The people who do comment on this possibility have usually been students who were told in advance what this work might lead to. More than we would like, people have concentrated on the information in the program rather than thought about the possibilities of this kind of information exchange. "Concerning this computer, I hope we never go into teaching children by this means even though I can see where it may be more efficient. Learning how to deal with people seems much more important to me." "The greatest difficulty here is that as well explained as the process is, it is still very confusing. I am not sure you could ever get the general public to use them as the general public strikes me as being very lazy and therefore would not consider this worth the effort." "I think the idea is very interesting and has many possible applications especially in finding out what the common level of awareness is and where work is needed to bring what area up." "A great start at showing the inter-relationship of various particular choices."

FURTHER NOTES ON SOCIAL IMPLICATIONS

Not all of the analysis which can be performed on the data from these demonstrations has yet been carried out. But perhaps the preceding discussion is adequate evidence that the earlier and following speculations and

philosophical musings are based on experiences with operational though still elementary prototypes.***

The practicality of public discussions

The data presented here is useful in estimating the feasibility of widespread use of "citizen sampling simulations" to involve the public in the planning process. It is to be expected that some people will be intimidated by the thought of using a computer and will be overwhelmed by this advanced communications technology. As was mentioned earlier, there is some evidence that a few people do have great difficulty with the program or at least proceed very slowly. Nevertheless, learning how to use a new technology and also exploring a list of 35 unusual social and technological developments is a demanding task for a one hour period. The success of these people in performing that task leads us to believe that community issues can be discussed by the public with this kind of technology, particularly as people gain practice in using it.¹⁰

The power of suggestion

The method of describing the future using information units seems to have been successful. Most people find the experience educational, and the basic structure of the program—the way the future is described—has not been criticized by most people who play the game. However, a few perceptive observers have expressed skepticism about the exploration, and rightly so. Simply listing a set of possible developments structures the thinking of explorers and thereby severely limits the range of responses. This is confirmed by the fact that people rarely suggest new developments or criticize the list given.

Those observers who are equally concerned but less theoretically inclined question the assumptions about the world which led to the selection of that particular set of forty information units. Furthermore there is a danger that some individuals may assume the changes in probabilities are "real" or indicate something more than merely the aggregated judgment of a group of individuals.

In future explorations we intend to make even more explicit the fact that the original probabilities and the changes in probabilities of other developments which we call "secondary effects" merely reflect the judgment

of the programmer and the people he has consulted and that the consequences indicated are not determined by a computer model based upon verified theories of the operation of social systems.

A program now being developed on a specific problem area, the Future of the University, uses probability, desirability, and importance ratings and both occurrence and non-occurrence matrices for three separate groups—students, faculty members, and administrators. In the Delphi program the computer is serving as a communications medium between the programmer and the people at the terminals. In the Future of the University program the fact that the computer is simply operating as a mediator among groups of people with different patterns of concern and perceptions of the world is made much more explicit.

Who is communicating with whom?

Even though the responses of each individual are not automatically seen by the other participants and the program itself may be changed only at intervals involving weeks or months, communication can still be taking place. In order to clarify the differences among Delphi-like computer programs it is useful to keep in mind whether individuals or groups are communicating with each other, the number of times a single individual will sit down at a computer terminal in order to work on one particular problem, and how much monitoring or editing of responses is done by the programming staff.

1. In Turoff's Delphi Conferencing, as I understand it, the responses of each individual are seen by all the other participants in the exercise. Items are added or dropped on the basis of a vote taken among participants. No monitoring or editing of the exercise is done by anyone except the participants themselves. Each time a person responds to a question or types a statement his response is not only recorded for viewing by the programmer but actually alters the program which each participant will view thereafter. Each person works on the given problem for a few minutes every day for several days.
2. The idea behind the Delphi Exploration, which can be thought of as a prototype "citizen sampling simulation," was to have communication take place not among individuals but rather between the planning group and the public. The responses of the explorers are recorded and viewed by the monitoring group but do not automatically change the program itself. This

*** An earlier consideration of possible social, political, and psychological implications of citizen participation in planning using computer-based communications media is contained in Reference 4.

pattern is similar to the normal process of instruction where communication takes place primarily between the teacher and the students. Each explorer works on a particular problem probably only once, at a sitting lasting from one to two hours. If the issue is a recurring one, the program is changed only every few weeks or months when the programming staff has an opportunity to reconsider the issue and to change the program on the basis of the responses obtained since the last modification of the program. The purpose of this kind of exercise is not to generate a forecast or a set of policy alternatives but rather to reduce the amount of time spent by planners in presenting background information to interested citizens and to generate data from the public on the desirability of particular alternatives, the completeness of the set of alternatives considered, and the way in which "the problem" is defined.

3. For a "computer-based mediator," such as the program on the Future of the University, communication takes place neither among individuals nor between planners and the public with the planners also acting as monitors, but rather between conflicting interest groups with either no monitor or a neutral party acting as monitor/arbitrator. The responses of individuals are not seen by the other participants except in the mean responses of a group. The position of each group is not arrived at by negotiation or compromise within the group but rather results from averaging the views of individuals in the group. In

the present program the responses of an individual alter his group's estimates of probabilities, desirabilities, and causal relationships, but the list of information units can only be changed by the programmer.

The practice in Delphi Conferencing of allowing the participants themselves to add and drop information units is a very important capability which can be incorporated into both citizen sampling simulations and computer-based mediators.

Exchanging views vs. simulating complex systems

There is a tendency to confuse this project with the presently growing number of attempts to model complex social systems. If that were our purpose, we would be greatly hampered by the technology we are using. The PLATO system was designed to operate a large number of computer terminals simultaneously with each terminal being allocated a small amount of computer memory space. Our efforts are sufficiently similar to the modeling of complex systems that we have attempted to keep up with developments in that area in hopes that the form of the models would be applicable to our programs.

However, given the limitations of our equipment for doing that kind of work and given its uniqueness for doing the task for which it was designed, we believe that it would be most productive to spend our time developing the computer as a communications medium between people and as a device for helping less skilled people to articulate their mental models of how the world works. Since computer models of social systems inevitably embody the assumptions of the programmer about what the important variables are, the ability of less technically skilled groups to express their assumptions about important variables could be helpful in trying to achieve a balance of political influence.

TABLE VII—Group Communication Techniques

	Delphi Conference	Citizen Sampling Simulation	Computer-based Mediator
Participants	individuals	planning group and the public	interest groups 2 or more
Length of Interaction	minutes	1 to 2 hours	1 to 2 hours
Number of Interactions	several, usually 1 per day	usually only one	usually only one
Normal Mode	usually group control and no monitor	at present completely monitored	at present only list of items not modifiable

REFERENCES

- 1 C E OSGOOD S UMPLEBY
A computer-based exploration of alternative futures for mankind 2000
Mankind 2000 pp 346-359
Edited by Robert Jungk and John Galtung
London Allen & Unwin 1969
- 2 D ALPERT D L BITZER
Advances in computer-based education
Science Vol 167 pp 1582-1590 Mar 20 1970
- 3 D L BITZER D SKAPERDAS
PLATO IV: An economically viable large-scale computer-based education system
National Electronics Conference Chicago 1968

4 S UMPLEBY

Citizen sampling simulations: a method for involving the public in social planning
Policy Sciences Vol 1 No 3 pp 361-375 Fall 1970

5 S UMPLEBY

The delphi exploration: a computer-based system for obtaining subjective judgments on alternative futures
Social Implications of Science and Technology Report F-1 pp 34-51 University of Illinois Urbana-Champaign August 1969

6 G MYRDAL

Objectivity in social research
New York Pantheon Books 1969

7 A bibliography of Delphi studies is included in
M TUROFF

The design of a policy delphi
Technological Forecasting and Social Change Vol 2 No 2 1970

8 R THEOBALD J M SCOTT

Teg's 1994
mimeographed 5045 North 12th Street Phoenix Arizona 1969

9 An explanation of how the 40 information units were selected is given in the third progress report

V LAMONT S UMPLEBY

Forty information units for use in a computer-based exploration of the future
Social Implications of Science and Technology Report F-2 University of Illinois at Urbana-Champaign March 1970

10 An experiment using the PLATO system to discuss a local environmental issue is reported in

V LAMONT

New communications technologies and citizen participation in community planning
Computer-based Education Research Laboratory University of Illinois Urbana May 1971

INSIGHT—An interactive graphic instructional aid for systems analysis*

by M. J. MERRITT and R. SINCLAIR

University of Southern California
Los Angeles, California

INSIGHT (INstructional Systems Investigation GraphiC Tool) is an interactive graphics program which illustrates the basic concepts of systems analysis. The transfer functions, inputs, and parameters of a single loop control system are drawn on the graphic display. Numeric data entry, system modifications and control of the time domain and frequency domain analysis is performed using the graphics terminal's light pen. All communications are problem oriented and no previous computer experience is required.

INSIGHT provides a common instructional tool for all disciplines touching upon systems analysis, control theory, and differential equations. Such disciplines might include engineering, physics, mathematics, geology, and chemistry, to name but a few. The common link between all of these fields is the use of Laplace Transforms to describe linear components of possibly non-linear systems, represented as serially connected block elements, with and without feedback. Additional common factors fall into two classes:

1. Common Instructional Goals
 - (a) encourage use of analysis techniques to develop insight into the process under study
 - (b) relate frequency domain analysis to time domain performance
 - (c) provide feel and intuition for theoretical concepts
2. Common Analysis Techniques
 - (a) time domain measurements of step, ramp, sinusoidal, etc., responses
 - (b) phase plane trajectories
 - (c) frequency domain analysis: root locus, Bode, Nyquist, and describing functions

The instructional device which provides all of these features should also be a universal educational solvent.

* Supported by the National Institutes of Health under Grant No. GM 16197-03.

It should be suitable for classroom instruction, laboratory exercises and homework assignments (see Reference 2 for a more complete discussion).

The work of Melkanoff,⁴ Moe,⁵ and Calahan¹ clearly demonstrates the advantages of computer aided instruction and computer graphics. The INSIGHT program is a continuation of these efforts which utilizes interactive graphics to meet all of the goals listed above.

All communications between the program and the user take place at the display console. Selection of system components, specification of parameters, and selection of computational algorithms are all accomplished with the light pen. The computational services provided by INSIGHT include both time domain analysis (numeric solution of the system equations) and frequency domain analysis (root locus).

Computational results are displayed as they are computed. All graphs are scaled and rescaled automatically to fit within the plot area of the display.

INSIGHT offers a number of unique advantages:

1. Learning time is short, usually less than five minutes.
2. The program is autoinstructional and completely protected against inappropriate user commands.
3. Both setup and solution times are extremely short.
4. No previous computer experience is needed.
5. No programming of any kind is needed.

The Hardware

All computer programs are constrained by the computer hardware available. INSIGHT is no exception. The USC School of Engineering's System Simulation

Laboratory contains (in part) the following equipment:

1. IBM 360 Model 44 with 64K bytes (going to 128K bytes as this is written), high speed line printer, and four disk files.
2. Adage AGT-10 Graphic Display System with 8K words (30 bits) of memory, light pen, function switches, joy stick, teletype and magnetic tape.

All of the facilities of the Adage Graphic System are available to FORTRAN programmers through the AGNOS language.³ The AGNOS language contains simple FORTRAN callable subroutines for image generation and manipulation, and light pen hit processing.

Using INSIGHT

INSIGHT is used in much the same manner as one would use a pencil and paper, only much more conveniently. The pages of paper are represented by a sequence of graphic "pages" appearing at the graphic console. Each graphic page elicits new information from the user relating to the design and analysis of the system.

INSIGHT provides the skeleton framework—a single loop control system with an arbitrary input. The user fills out this framework by selecting items from a menu and inserting them in the control systems block diagram. The first graphic page presented to the user is shown in Figure 1.

The block diagram is the process, and vice versa. In order to reinforce this feeling, the block diagram is always retained at the top of the display. All other material is entered into the lower half of the screen.

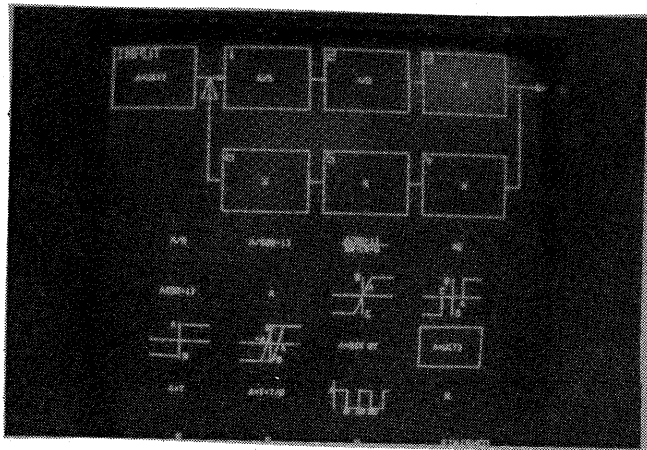


Figure 1—First graphic page of INSIGHT showing the block diagram and element menu

TABLE I

Linear Transfer Functions	Non-linear Functions	Forcing Functions
A	SATURATION	A SIN (Bt)
$\frac{A}{s}$	RELAY	A U(t)
$\frac{A}{(Bs+1)}$	BANG-BANG	At
$\frac{(As+1)}{s^2+BS+C}$	GEAR TRAIN BACKLASH	$\frac{1}{2}At^2$
As	CUBIC	SQUARE WAVE
A(Bs+1)	SAMPLE AND HOLD	GAUSSIAN RANDOM NOISE

D parameter is the initial condition for all linear transfer function elements

The contents of the element menu are summarized in Table I. All of the photographs seen in this article were taken from an early version of INSIGHT. Whenever a discrepancy exists between the text and the photographs, the text is correct.

Menu elements are placed in desired blocks in two steps:

1. Selection—when a menu item is touched with the light pen, INSIGHT affirms the touch by moving the square shown enclosing the item A/S in Figure 1, to enclose the touched item. This item is the "current selection."
2. Placement—when the contents of one of the control systems blocks is touched with the light pen, the current selection is copied into it, replacing its previous contents.

Initially, all of the control system blocks are filled with asterisks. An asterisk filled block will be treated as a unity gain element by all of the computational algorithms.

INSIGHT recognizes obvious errors, for example placing a forcing function, A SIN (BT), in a transfer function block, or the reverse, placing a transfer function element, A/S, in the INPUT block. When either of these placements is detected, the words "ILLEGAL OPERATION" are written at the top of the display. The contents of the block involved are not changed.

When the block diagram is complete, the user touches the word FINISHED with the light pen. The

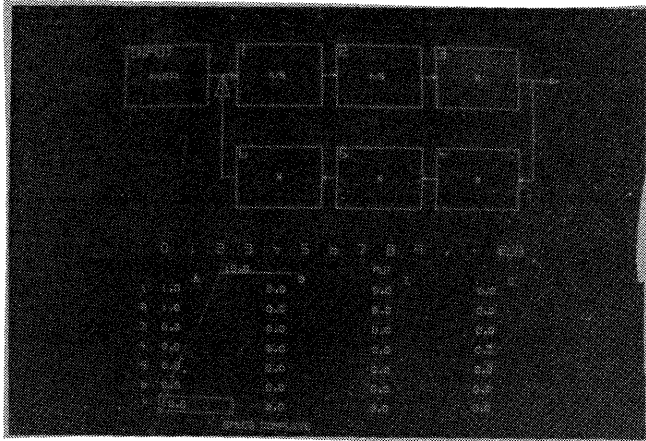


Figure 2—INSIGHT's specification page, parameter entry for the diagram shown is complete

element menu is removed, and the parameter specification page written in its place, see Figure 2.

Each block in the control system is assigned a reference number. The reference number is associated with a block's contents, as well as its output. The number 3 alone refers to the output of block 3. The INPUT block is assigned a reference number of 7. A given block may contain an element with one, two, three, or four parameters. These parameters are denoted by the letters A, B, C, and D (see Table I).

The parameter values are arranged in seven rows of four columns each, as shown in Figure 2. All parameter values are set to zero when the INSIGHT program is entered. Thereafter parameter values are retained until they are changed by the user.

Parameter entry is a three step process: (1) preparation, (2) selection, and (3) placement, as follows:

1. Preparation—a 13 item menu containing the numbers 0 through 9, decimal point, minus sign and RUB (backspace) is written below the block diagram. The underlined value seen just below the numbers 2, 3, and 4 in Figure 2 is called the temporary line. As menu items are touched, the corresponding character or backspace is written on the temporary line.
2. Selection—when a parameter value is touched by the light pen, the box shown enclosing the A parameter, in Figure 2, is moved to enclose the touched parameter. This parameter becomes the "current selection."
3. Placement—when the word PUT is touched with the light pen, the current contents of the

temporary line are copied into the box enclosing the current selection. The temporary line remains unchanged, allowing the same value to be stored in a number of places.

Integer values may be entered with and without a decimal point. For convenience in entering complementary numeric constants (for relays, saturation, etc.), the minus sign may precede or follow the numeric characters, thus -1 and $1-$ are both stored as -1.0 .

When all numeric values have been entered, the words "SPECS COMPLETE" are touched with the light pen. The parameter specification page is replaced with the mode specification page, shown in Figure 3.

The mode specification page allows the user to select between the two computational algorithms:

1. Time domain solution
2. Root locus

(NOTE: These options are not shown in Figure 3, which was taken from a preliminary version of INSIGHT.)

If the root locus option is selected, INSIGHT adds an auxiliary gain K to the control systems open loop transfer function. Pole and zero positions are marked on the root locus plot by Xs and Os respectively. Root positions are plotted for values of K in the range zero to ten. The root positions at $K=1$ are the roots of the control systems characteristic equation and are denoted by small squares instead of the usual asterisk, see Figure 7. The locii are drawn as they are computed and are automatically rescaled to fit into the plot area of the display. Four options are available while the locii are being drawn: STOP, START, RESTART, DONE.

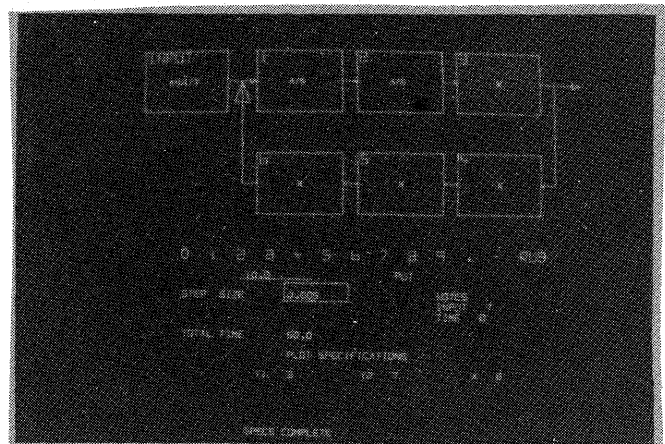


Figure 3—Mode specification page of INSIGHT—Showing default parameter values

TABLE II—Graph Formats

X	FORMAT	KEEP
0	phase plane Y1 versus Y2	NO
1	same as X = 0	YES
7	time histories Y1 and Y2 versus time (see Fig. 4)	NO
8	same as X = 7	YES

The STOP and START options are normally used to freeze the display for purposes of discussion or reproduction. If, at the end of a locus, i.e., $K=10$, START is touched, then the locus is continued to $K=20, 30$, etc. The decision to STOP a root locus cannot be made effectively until the entire locus has been viewed. The RESTART option allows the user to return to the beginning of the sketch and STOP it at selected intermediate positions. Touching DONE with the light pen returns control to the first graphic page.

The root locus algorithm treats all non-linearities as unity gain elements. If describing function gains are known, they must be inserted in place of the non-linearities before requesting the root locus analysis.

If time domain solutions are requested, then the five parameter values, step size, total time, Y1, Y2 and X, are examined. If the step size is too small, i.e., if

$$\text{total time/step size}/> 100,000$$

then the step size is set to

$$\text{step size} = \text{total time}/100,000.$$

Time domain solutions may be displayed in either of two formats, with and without a KEEP option. In all cases, the reference numbers placed next to Y1 and Y2 specify the two block outputs to be plotted. The reference number placed next to the X symbol in the mode specification page determines the format of the graph, as shown in Table II.

If the KEEP option is selected, the current solution is drawn on top of the previous solution or solutions at the same scale factors. The KEEP option is useful in modeling and optimization studies and demonstration of the properties of phase plane singularities.

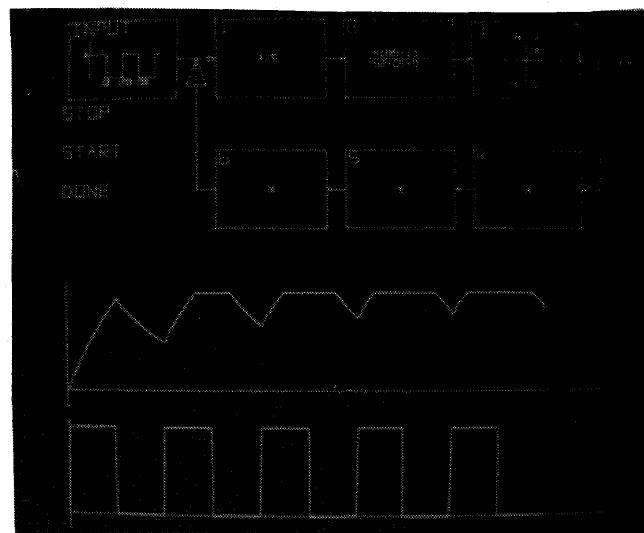


Figure 4—Non-linear control system with saturation and a square wave forcing function

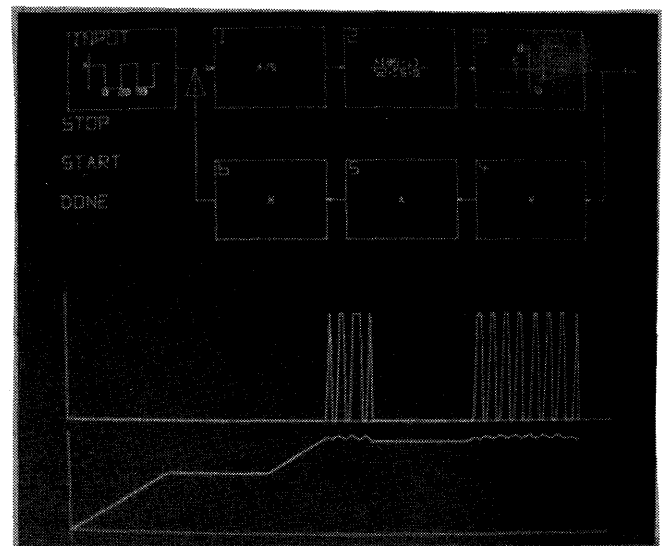


Figure 5—Non-linear system with relay element in place of saturation element

Values are added to the graphs as they are computed. Scale factors are adjusted automatically to fit the graph in the plot area (unless the KEEP option was selected). The STOP, START, and DONE options halt and resume computations and terminate the solution, returning control to the first graphic page.

Application of INSIGHT

An instructor wishes to demonstrate the effect of non-linearities on system performance and to illustrate the concepts of limit cycling and stability. He begins by constructing the control system shown in Figure 4.

This system contains three poles and one zero in its open loop transfer function. Numeric values must be

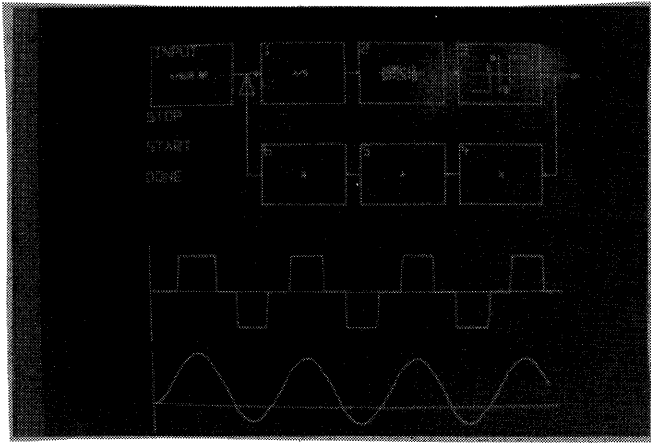


Figure 6—Non-linear system with sinusoidal forcing function

specified for the gains, time constants, slope of the linear segment of the saturation element, and for the saturation levels. After specifying the integration step size and total solution time, the instructor selected the time domain solution, plotting the output of blocks 3 and 7 versus time.

In Figure 5, the instructor has replaced the saturation element with a relay and plotted blocks 3 and 1 versus time.

In Figure 6, the instructor has switched to a sine wave forcing function and has plotted the input to, and output from, the relay element versus time. This could be used to introduce describing function analysis.

The ease of use, fast solution time and rapid interaction of INSIGHT are illustrated by the fact that less than five minutes was required to generate the three examples just described.

INSIGHT may be used for its root locus facilities alone. Another instructor, desiring to discuss properties of the root locus sketch, selected elements to form an open loop transfer function of the form:

$$G(s) = \frac{60(0.0833 S + 1)(0.2 S + 1)}{S(S^2 + 4S + 20)}$$

The resultant INSIGHT root locus diagram (again from an early version without axis labels, magnitudes or $K = 1$ markers) is shown in Figure 7.

FUTURE PLANS FOR INSIGHT

The single loop structure and restriction to seven blocks are major disadvantages. They are, however, removable through additional programming. The high cost of the dedicated IBM 360/44 and the Adage dis-

play system is not so easily treated. Calahan¹ and others have discussed the cost factors relating to the use of computer aided instruction. A number of satisfactory graphics systems are available for less than \$10,000 (Techtronics T4002 and the Adage ARDS System, to name only two). The low cost of these devices, combined with increased demand for computer time will encourage administrators to restructure the financial foundations of educational computer centers. Conversion of INSIGHT to operate within an existing Conversational Programming System (CPS) utilizing a large screen video device is under study. The decrease in display capacity and increased response times will be offset by the decreased cost and increased availability.

SUMMARY AND CONCLUSIONS

INSIGHT provides a variety of Systems Analysis functions to a wide range of users in a convenient, easy to use package. The goals set forth at the beginning of this article were met and exceeded. Extension of INSIGHT's capacities to encompass more complex structures and additional analysis tools is in progress.

ACKNOWLEDGMENTS

The authors are indebted to Rick Klement, Bill Liles, and Al Vreeland for the AGNOS language, and to

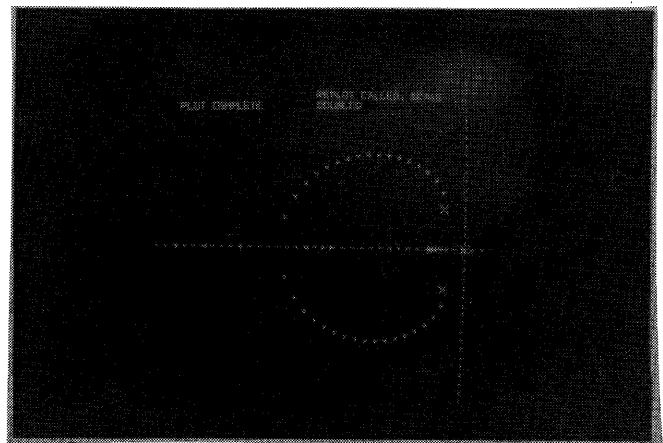


Figure 7—Root locus diagram for a three pole two zero open transfer function

Donald Miller for his many valuable suggestions in the development of the INSIGHT program.

REFERENCES

- 1 D A CALAHAN
Circuit design application of the Michigan terminal system
IEEE Transactions on Education Vol E-12 No 3
September 1969
- 2 M L DERTOUZOS
Educational uses of on-line circuit design
IEEE Transactions on Education Vol E-12 No 3
September 1969
- 3 R KLEMENT W LILES M MERRITT
A VREELAND
The AGNOS language
University of Southern California Technical Report
No 71-19 April 1971
- 4 M A MELKANOFF
The use of on-line graphical computer systems for student research
IEEE Transactions on Education Vol E-12 No 3
September 1969
- 5 M L MOE
The NASAP computer-aided circuit design program and its use in undergraduate education
IEEE Transactions on Education Vol E-12 No 4
December 1969

An interactive class oriented dynamic graphic display system using a hybrid computer

by A. A. FRANK

University of Wisconsin
Madison, Wisconsin

INTRODUCTION

The current state of engineering education is such that it is often difficult for students to gain insight into the many subjects they must master. To aid the professor or teacher in "getting the point" across, the following system is being tried.

In every field of engineering, theory is developed using mathematical and graphical techniques. These theories are manifested in problems which illustrate different aspects of the theory. These problems in general are rather tedious to work out by hand and when they are worked out they illustrate one single aspect of the theory. For example, a beam in a strength of materials class has a given load, then how does the shear, moment and displacement change for different kinds of loading? What effect does the cross-sectional inertia have? How does cross-sectional inertia change as the shape is changed? What are the natural frequencies of this beam? How would it bend if a natural frequency were excited, etc? This kind of problem could easily be solved by a hybrid computer system and the solution displayed on an oscilloscope screen. The answers to all these questions and many more can be easily seen in a continuous fashion on command of the professor.

It must be emphasized that this system is not intended to be a student involved teaching aid.^{1,2} It differs with other elaborate systems used for specific applications in that the machine and language remains general and only the "skilled" operates the machine.^{3,4} Further, it is possible to display "static" as well as "dynamic" problems as illustrated by the example.

To make the system worthwhile and utilize the hybrid system more effectively, it is necessary to consider multiple numbers of terminals placed into different classrooms. It is then necessary to solve each classroom's problem on a shared basis. A hybrid system with a digital computer which has a real time monitor and a disc or drum storage can be used in this mode.

Then since the digital computer can control fully the analog computer, dynamic displays in real time can easily be generated.

To make this system work in a reasonable manner in a university environment it is an absolute necessity that the professor not be burdened with the additional responsibility of writing his own programs. To solve this dilemma a full time staff member whose sole duty is the programming of class problems is provided. This staff member is the key to the success or failure of the system. He must be versatile enough to handle all fields of engineering.

It must be emphasized that the professor is not to be replaced by the system, but rather the system is to provide him a more effective manner in which to teach, or in other words to enable courses to contain more material for a given amount of time, and to provide the student a means to comprehend this material more effectively. It is thus the responsibility of the professor to learn about this new teaching tool.

DISCUSSION

I. System Concept

The hybrid computer is the heart of the display system. A normal hybrid computer system has the elements shown in Figure 1.

The computer system has all the elements of a display computer. The objective is to design a system which can maintain a display system without greatly jeopardizing normal hybrid computer usage.

In a normal environment the hybrid computer's digital machine experiences only about 5-10 percent CPU usage. The graphical display will essentially steal from the unused time. It has plenty of time from which to pick. Besides, there is a full time operator so communication between operators is possible.

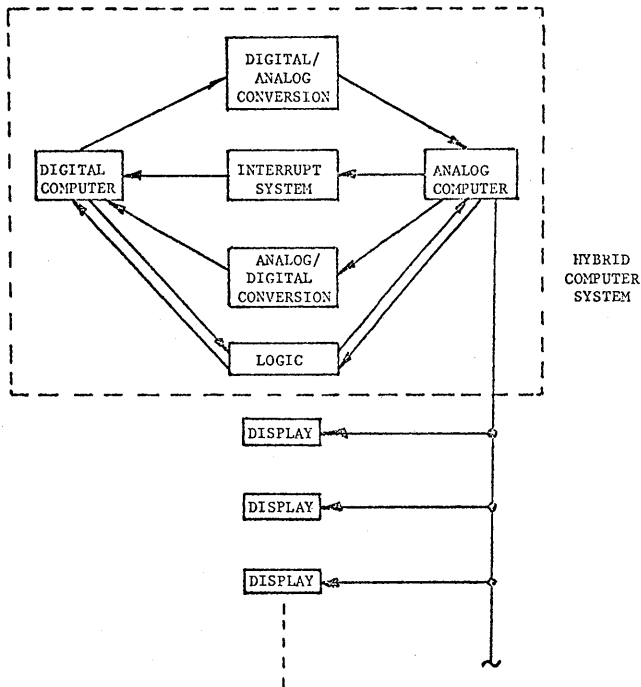


Figure 1

The full time operator's duties include both machine set up and operation, and programming advice.

HARDWARE

The hybrid computer for such a system must consist of at least the following complement of equipment:

Digital Computer

- 1—General Purpose CPU.
- 2—16 K Core memory.
- 3—Disc or drum mass storage.
- 4—Real time monitor system.
- 5—Priority interrupt structure.
- 6—External communications: A/D, D/A, logic in and out, and hardware interrupts.

Analog Computer

- 1—General purpose computer with patchboards.
- 2—16-64 integrators.
- 3—4-40 multipliers.
- 4—30-60 summers.
- 5—100-200 digital computer controllable potentiometers.
- 6—Lines to various display centers.
- 7—Hard copy capability.

Each of the displays in the system consists of the

following:

- 1. 8½×11 memo scope or scope with memory.
- 2. Keyboard.
- 3. 8 Coefficient potentiometers.

SYSTEM OPERATION

The organization, hardware and software, is designed to provide an aid to the teaching of engineering principles for the professor. Thus the display of a problem solution must be available immediately (within a second) after the professor's request is put into the system. To do this, the hybrid computer is time shared with the various terminals and the user of the hybrid computer.

The operation of a terminal is done by a professor simply pushing a request button. On initiation by the professor, his program will be presented with the set of parameters he has specified on the display screen in less than one second. The professor can then change the coefficients or parameters and push the request button again and the students can see within a second the next solution to the problem. These solutions can be stored for comparisons or the screen can be wiped clean each time. This is the feature of the memory screens. If a "hard copy" is desired the professor need only to call the operator and make the request. The operator then will use the hard copier and provide the professor and class with hard copies. These hard copies could also have been made in advance in which low cost reproduction methods would be employed.

The professor may have a number of problems programmed into the computer system. He can simply call for the problem by name through the keyboard. For example, the bending beam problem can be broken into at least three separate parts; (1) shear-moment diagrams, (2) deflection-stress diagrams, (3) vibration mode diagrams, etc.

The development of the software for each problem is the responsibility of the display systems personnel. The only responsibility the professor has is to specify the problem and the parameters he would like to see varied, and the time and date the problem will be requested. His problem will be stored in the computer. Of course, it will be easy for a professor to specify a problem which is beyond the computers capability. This will either be due to the problem magnitude and the computer capacity or due to the fact that the problem is not suitable for this kind of computation. It is not likely that such problems, at the present level of undergraduate education, will be encountered; however, to provide for

such eventuality the terminals will be outfitted with time share capabilities to other larger facilities, such as a large IBM 360 system or an 1108 Univac system via telephone connections.

The program developed by the systems analyst will be put onto the digital computer's mass storage device (either a disc or a drum). In this fashion, when the professor makes a particular problem request the digital computer fetches the program from the disc or drum and executes it and sends the output to the display. In problems involving the analog computer, or real time dynamics, the digital computer provides the control and direction of the analog computer and its signals. Problems involving the analog computer will require a patchboard as well as the digital program. It is the responsibility of the system staff member to see that such a patchboard and the digital program be on the machine and ready to run when the request button is pushed. The professor may use this display for only ten minutes during his lecture and may only use it three or four times during the semester. Even with 30 professors throughout a college using such a system, it will have a relatively low demand if a little care is taken in scheduling.

SOFTWARE SUPPORT

The staff must be capable of providing this service to any professor and any department in the engineering college. For example a professor in the engineering graphics area wishes to present a problem in descriptive geometry, such as a cone cutting a cylinder, on the display system. Another example may come from the chemical engineering department which has a problem involving chemical kinetics and process control. Still a third example may come from civil engineering in the design of earthquake proof structures. Obviously every

department of engineering has courses which can find uses for such a system. The important aspect of the total system is that the staff must be able to comprehend all areas of engineering so as to aid professors from all disciplines. The program can be called upon once a semester or whenever such a class is run.

As the system becomes used by the College of Engineering the facility may have to be expanded to a separately dedicated computer system. However, a way to begin such a program is to initially start with an existing hybrid computer system as an overload. Then when the system has proven itself and grows beyond the hybrid labs capability it will be easy to justify further expenditures and argue for its own system and staff.

Most important is that the professor's present teaching techniques need not be modified by any great extent and thus this system lends itself to a natural "phase in" period. It is but a stepping stone to more elaborate systems.

REFERENCES

- 1 J LENAHAN
Synthesis of an interactive human-machine system
PhD Thesis University of Wisconsin 1969
- 2 F KOENIG
Formal analysis for a general system of interactive automata
Presented at the 3rd Hawaii International Conference on System Sciences January 1970
- 3 D CALAHAN
Circuit design application of the Michigan terminal system
IEEE Transaction on Education Vol E-12 No 3
September 1969
- 4 M MELKANOFF
The use of on-line graphical computer systems for student research
IEEE Transactions on Education Vol E-12 No 3
September 1969

Hybrid terminal system for simulation in science education

by DONALD C. MARTIN

North Carolina State University
Raleigh, North Carolina

ONCE UPON A TIME ---

Once upon a time . . . there was an old woman who lived in a shoe. She had so many children, she didn't know what to do. . . .

. . . Precisely the problem which faces the professor who wishes to use computer simulation in education today. Like the old woman, he has no serious difficulty with the older children who are helping around the house or in graduate school, but what can be accomplished with the thousands of youngsters now lodged in or entering that shoe we call the university. This paper is a progress report on a new approach taken by one school to introduce continuous system simulation concepts to all four thousand of its children.

Once upon a time . . . we attempted to teach analog computer programming in various engineering and science oriented courses. The majority of students objected, and well they should. Ten years ago, they objected by writing short references to the professor on the desks or bathroom walls. Today, they simply ask in class—where will I use the analog computer after I enter the real world. The answer is, of course, that ninety-five percent or more will never see a general purpose analog computer after they graduate. They all need to know about analog and digital simulation, operational amplifiers, basic electronics and signal conditioning, but the majority will never require the ability to patch a six degree of freedom or nuclear reactor simulation. Wouldn't it be nice if they could all use such simulations to study complex systems response without the patching exercise?

Once upon a time . . . it seemed that digital simulation would be the answer to the old patching and scaling problem. Once students grasped the idea of implicit or bootstrap solution of differential equations, they could easily learn a digital simulation language structure in several hours. The scaling problem largely disappears and the output can be displayed on an oscilloscope or plotted. The difficulty with most digital simulation

languages is speed and availability. It is far too expensive for most universities to provide sufficient remote terminals to effectively service an engineering school of several thousand students. One terminal for every fifty students is not unrealistic and even this number will lead to long hours waiting for access during prime time. The time shared terminals currently being developed by Grannino Korn¹ in project DARE at the university of Arizona are certainly encouraging and may change this picture in the future. At any rate, it is fair to say that the high cost of interactive digital terminals for continuous system simulation severely restricts their extensive use in the undergraduate science education program today.

Once upon a time . . . students were asked—no, required—to submit lengthy laboratory reports to be read and graded by the graduate teaching assistant. The hours of hand calculations were gradually superseded by reams of computer output. It would appear that much of the undergraduate laboratory could be improved by using simulation techniques and programmed instructional material. The student would be required to answer specific questions about the physical system, either real or simulated, to demonstrate his acquisition of the information presented in the laboratory.

Now . . . It is because of the nature of interactive simulation and its use as an aid to the students' understanding of physical processes that the hybrid terminal system described in the remainder of this report was developed. An ideal system includes the high speed, interactive, graphic display capability of the analog computer along with the memory, program storage and terminal capability of the digital computer. The design parameters for such a terminal were recently described.² These hybrid terminals evolved from earlier work with student evaluation of simple and inexpensive analog computer terminals.³ The final terminal design was based on the premise that the student need not learn analog patching to use the hybrid terminal and that

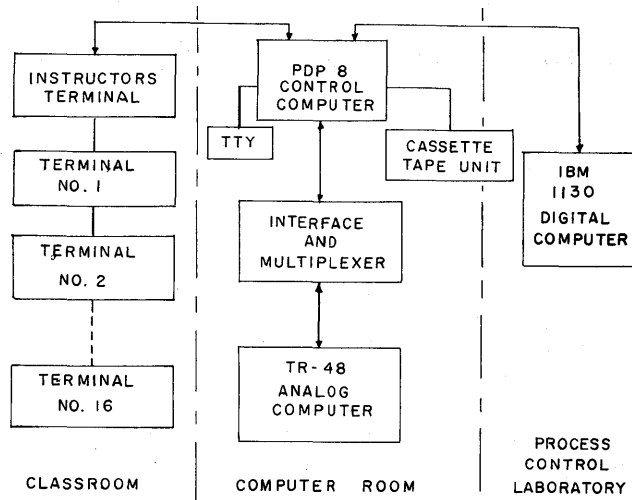


Figure 1—Hybrid terminal system

programmed instruction type laboratory handouts should replace the traditional concept of submitting hard copy computer results of a simulation.

THE HYBRID TERMINAL SYSTEM

The hybrid terminal system developed at North Carolina State University is the largest system in the world devoted to undergraduate system simulation utilizing both analog and digital computers. This system, funded by the National Science Foundation, was designed and installed in less than twelve months. The hardware was supplied by Electronic Associates and the software was developed by student programmers at the university.

A flow sheet for the classroom hybrid terminal system is given in Figure 1. The system consists of the following components:

- a. Sixteen student simulation terminals.
- b. An instructor's control terminal.
- c. A digital mini-computer with teletype and cassette tape I/O.
- d. Control interface to the analog computer.
- e. Channel communication link with an IBM 1130 for hybrid problems.

The student simulation terminals

The basic terminal configuration is shown in Figure 2. All display functions are located on the upper display panel. Control and data input are provided on the in-

clined panel. The control functions available to the student include power on-off, store, non-store, and erase. Indicator lights are also provided for terminal identification, error and terminal ready status.

The primary output display device is a Tektronix type 601 storage screen oscilloscope. Output scaling is automatically provided for in the digital software. A six digit display is available to return parameter values and other numbers from the digital control computer.

The student controls the analog or digital simulation from the keyboard. There is provision for selecting from two X and four Y channels for the display of analog signals from any of four pre-patched analog problems. The student sets any of eight function switches and enters values for up to eight parameters from this keyboard. The current value of any parameter can be displayed in the six digit window on command. Either E or F format can be selected by setting a two position switch. Hybrid problems are all addressed as problem five, i.e., selecting problem 52 is for digital simulation, problem 51 for least squares fit of data, etc. Provision is made on the terminal for automatically incrementing a parameter through a range of values and for control of a cursor to locate points on the analog display.

Instructor's control terminal

The instructor has access to a terminal which is similar to the students as shown in Figure 3. The major difference lies in the output display. This terminal uses a Tektronix type 4501 storage oscilloscope with television



Figure 2—Student simulation terminal

output. The instructor can display his solution to the entire class on a large screen television monitor and mix visuals with the computer output. By selecting the appropriate terminal number, he can also display any student's solution on the monitor. He also has the capability of obtaining a hard copy of his or a student's solution of any given problem in ten seconds with a Tektronix type 4601 hard copy unit.

Digital control computer

The heart of the hybrid terminal system is the digital control computer. This is a PDP-8 with 4K core and cassette tape drive for program storage. This computer collects and stores data from the simulation terminals until a solution is requested by a student. When a solution request is received for an analog problem, appropriate problem number, outputs and function switch settings are transferred to the multiplexer. All parameters are normalized, the digital to analog converter set, and the analog is placed in the operate mode. Only the oscilloscope of the terminal which initiated the solution request is unblanked. The basic cycle time for this process is forty milliseconds although solution time can be extended by the instructor if desired.

If a hybrid problem is selected by a student, the PDP-8 computer interrupts the IBM 1130 and initiates the transfer of the appropriate program from disk to core. All further entries from this particular terminal are then transferred to the IBM 1130 until the hybrid program has been executed or terminated by the

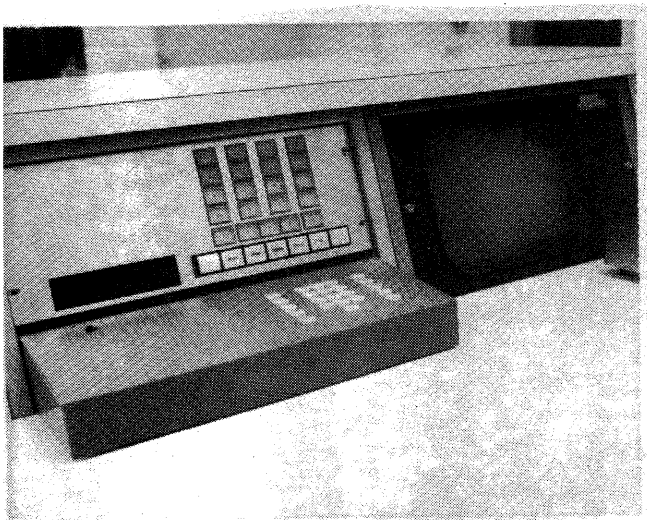


Figure 3—Instructor control terminal

student. For a hybrid or digital simulation program the IBM 1130 stacks interrupts and stores input on the disk for sequential execution. After execution, the IBM 1130 interrupts the PDP-8 control computer which steers the output back to the proper terminal.

The instructor uses a special conversational language developed by our programmers for setting up student application programs. This language makes use of the cassette tape recorder to store maximum and minimum values of parameters, analog operate and reset time, and output display scaling information. The only expertise required by the instructor who wishes to employ this system in his course is basic analog computer programming and a knowledge of FORTRAN for digital applications.

Control interface

The interface couples the PDP-8 control computer with the analog computer multiplexer, and terminals. If the student wishes to enter a parameter value, this information is transferred digitally from a 32 bit shift register in his terminal to his core area in the PDP-8 through this interface. If a parameter display is requested, values are transferred through the interface back to the display window of the terminal. A solution request directs the interface logic to transfer all information collected from a given terminal to the analog computer. The interface clock determines basic cycle time for analog problems. This interface also contains the logic for interrupt processing between the PDP-8 and IBM 1130 computers for hybrid or digital problems.

Channel communication link

Since the PDP-8 control computer only has a 4K core, hybrid and digital problems were extremely limited on the basic system. To implement digital simulation and more extensive hybrid problems, the PDP-8 was coupled to an IBM 1130 located in a nearby process control laboratory. Each computer treats the other as an additional device operating under interrupt control. A basic monitor was written for the IBM 1130 to transfer terminal information to disk and call stored digital programs.

TERMINAL CLASSROOM

The sixteen terminals are installed in a classroom near the computer room as shown in Figure 4. Three terminals are located on a large conference table near the instructor's console. These are used for small groups of

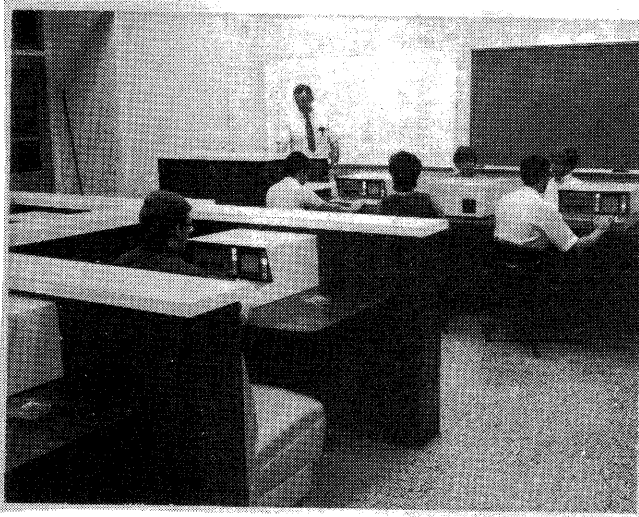


Figure 4—Terminal classroom

students in a seminar-discussion mode to provide a high degree of interaction with the instructor. The remaining terminals are located in restaurant-type booths designed for use by two students. The classroom can handle thirty students comfortably. The terminals are series connected with the last terminal of the string located approximately 350 feet from the computer room.

SYSTEM RESPONSE TIME

The basic system response time for analog programs is 40 milliseconds. This time is limited by the old but reliable relay mode control analog computer used in our system. When a student requests a solution after selecting a problem and setting parameters, the next twenty milliseconds are used to set the multiplexer, set any or all of the eight DA converters to his parameter values and reset initial conditions on the analog computer. During the next twenty millisecond time span, his display oscilloscope is unblanked to store or view the problem solution curves. In this mode of operation, the worst case response time when all sixteen students request a solution simultaneously is 640 milliseconds.

The response time for hybrid problems and digital simulation is naturally dependent on the specific application. A typical illustration is the least squares data program described in the next section. This program requires from thirty seconds to one minute execution time on the IBM 1130, the time dependent on the order of polynomial requested. Thus, if two or three terminals are using this program, an individual would have to wait several minutes for a solution. Since the communication channel operates on an interrupt

basis, the analog solution response time for other terminal users would not be degraded. It should be noted that only one hybrid or digital simulation problem can be entered and executed from an individual terminal at any given time. Any attempt to enter a second hybrid problem before completion or deletion of the first results in an error message on the screen. This error message directs the new user to a terminal which is not busy. This feature is particularly important when using the digital simulation program which can result in individual terminal response times of 20-30 minutes for multiple users. In such instances, the user can leave the terminal and return at a later time to request his solution. Again, analog solution time is not affected on the remaining terminals.

TYPICAL PROBLEMS IMPLEMENTED ON THE SYSTEM

The types of problems which can be implemented on the hybrid terminal system are illustrated with three specific examples. The first is an analog water pollution study used at the freshman and sophomore level and the second is a digital data reduction problem used by juniors and seniors to analyze laboratory data. The third example illustrates control features of the digital simulation language.

An analog problem

This problem demonstrates the effect of dumping pollutants in a stream at two different points. The

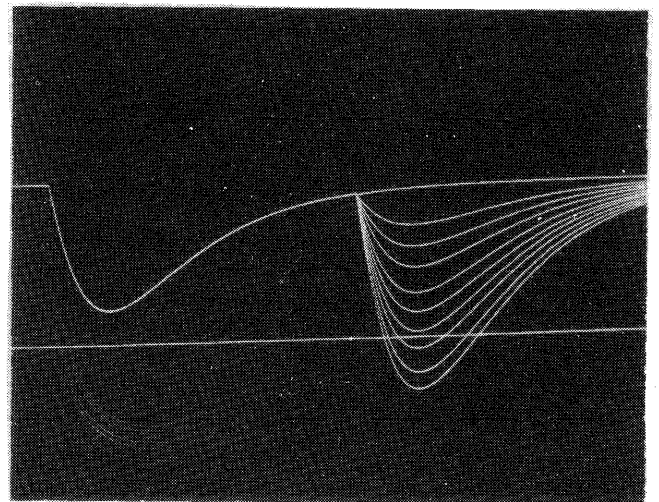


Figure 5—Typical output for an analog problem

primary display presented to the student is the typical oxygen sag curve, but he can also look at the waste decay function. The model consists of four first order differential equations, two for each town on the stream. The student can control the decay rates characteristic of the waste, the rate of dumping waste material, and the reaeration coefficient as a function of stream velocity and turbulence. He is given a programmed instruction type handout, and asked to determine from the terminal, waste characteristics, rates, and stream aeration coefficients so that the oxygen level will remain above habitable levels for fish population. This problem is designed for a two hour laboratory session. Since this is an all analog simulation, response time is excellent. A typical illustration of the parameter increment features of the terminal is shown in Figure 5. In this case the student has incremented the quantity of waste dumped at the second town from its minimum to maximum value. For single student operation, this plot is obtained in 400 milliseconds. If all were requesting ten solutions of this or another analog problem at the same time, the plot would have been obtained in about seven seconds. Student feedback from this problem has been very favorable, partly because of the current nature of the problem.

A digital problem

A polynomial fit of experimental data serves to illustrate the terminal system capability in the digital mode. When the student selects any problem number beginning with the numeral five, his terminal is coupled

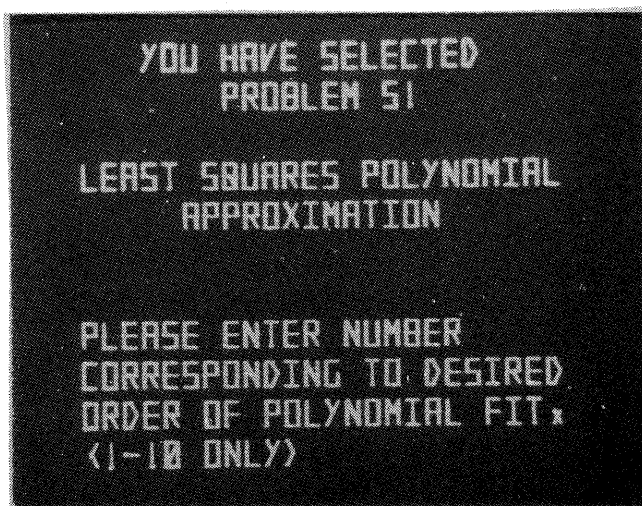


Figure 6—Initialization of the least squares program

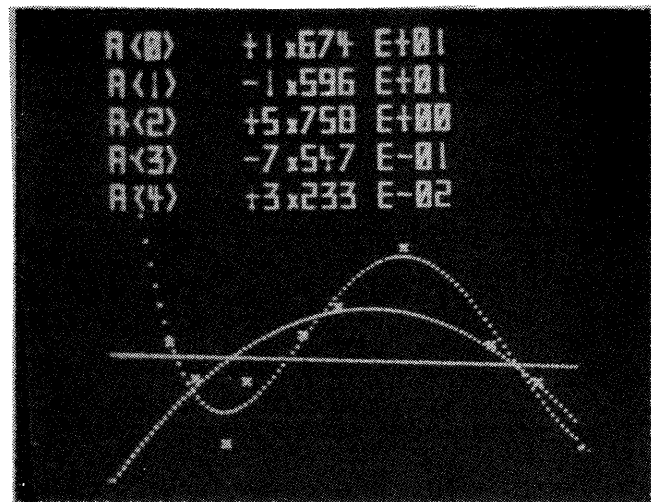


Figure 7—Output of digital curve fitting problem

to the IBM 1130 through the communication link. Figure 6 shows the response received at the terminal for the least squares program, number 51. The student can then either select card input for data or enter the data on the terminal keyboard in XY pairs. He then specifies the order of the polynomial and requests a solution. The data, curve and coefficients are returned as shown in Figure 7 in approximately one minute. For this particular plot, the student requested a first, second, and fourth order polynomial and then requested the coefficients for the fourth order case. The user scales the output by entering the percentage of the vertical screen he wishes the input data to occupy. Naturally the response time would be on the order of sixteen minutes if all terminals requested a solution simultaneously, but such use is unrealistic with this type of program.

A digital simulation problem

The digital simulation program is accessed in the same manner as other digital programs, i.e., selection of problem 52 provides the terminal response shown in Figure 8. The student can enter configuration statements, parameters, and control statements as indicated. Incidentally, the graphics package written by our student programmers requires about 400 milliseconds to fill the screen. Programs which exceed the line capability of the display are paged. Since the IBM 1130 simulation language is slow, between two and eight minutes are required for digital execution time. The user's program is tagged and he can return later to

```

ENTER CONFIGURATION DATA
  INT, 1, 2, -3, 0, 3, 05
  SUM, 16, -4, 22
  LAST
NO CONFIGURATION ERRORS
ENTER PLOT VARIABLES
  X1, 0 x 0, 50 x 0
  Y3, -10 x 0, 20 x 0
READY
RUN

```

Figure 8—Digital simulation control

obtain his output if desired. Another user can solve an analog problem from this terminal, but if he attempts to enter a digital or hybrid program, a message on the screen directs him to another terminal which is not busy. In this case, a busy terminal is one which has any hybrid solution pending. This problem could be avoided with a larger digital computer with additional storage capability.

CLASSROOM EXPERIENCE

At the time of this writing, the system has been installed and used for one semester. Approximately one thousand students have used the system during this initial phase of operation. These students were from the Chemical Engineering, Engineering Operations, Civil Engineering, and Computer Science Departments. Many others have expressed interest in using the system and will be developing application programs during the summer. One interesting use has been by the Freshman Engineering Division. These students take an introductory engineering orientation course and have no background in differential equations or system simulations. They seemed to have little or no difficulty in understanding the concepts presented in the study of a simple stream pollution simulation.

The time required for a student to learn the operation of these terminals is about two hours. Most of this time is spent in overcoming a natural reluctance to press the buttons without detailed instructions concerning their function. Once they decide for themselves, for instance, that a value entered on the keyboard will not alter a parameter value unless the ENTER key is used, they

seldom have further hesitation. We try to ensure that the first two problems used by a given group are of the programmed instruction type with very detailed instructions on terminal operation. Subsequent handouts generally include detailed description of the physical system being studied but not much information on terminal operation.

Typical problems used in an introductory chemical engineering systems analysis course include:

Session one: An introductory session used to teach terminal operation. The example used is the filling of tanks of various sizes with different fluid flow rates.

Session two: A perfectly mixed tank is forced with a step function and sine wave to illustrate superposition for linear systems. The student controls the tank volume, inflow rate, input frequency, and initial concentration. He uses the simulation to reinforce the concept of time constants and determines phase lags and amplitude ratios for sinusoidal forcing of a linear system.

Session three: The student studies the response of first order systems in series from the oxygen sag curves of a two town river pollution model.

Session four: A mercury manometer is forced with a step function and the student observes the actual response in terms of frequency and damping. He then uses the terminal with a second order model to see how closely calculated values of the parameters in the simulation fit the experimental data.

Session five: A heated tank model is studied to introduce the idea of proportional, integral and derivative control. The student varies control parameters to demonstrate the idea of stability of such control systems.

Additional laboratory sessions introduce frequency response, sampled data systems, etc. The laboratory report consists of answering questions related to system response as determined from the simulation.

STUDENT FEEDBACK

Initial feedback from the students using this system has been excellent. They are asked to comment on each experiment and hand in their comment sheet without identifying themselves. Approximately 80 percent are ecstatic and make such comments as:

"The most interesting lab in the university."

"I never understood stability until I actually saw the system response on the terminal."

"The labs improve with time because as you go on you become more accustomed to the machine. It was also easier to follow instructions for this third

experiment. They made more sense. Too bad we have to stop when we are barely started."

"Lab so far has been very interesting, and more important, useful. My only concern (sad but true) is how these reports will be graded."

As expected, a few adverse comments were received, i.e.,

"This lab can be instructive but not in two hours! The time to fully understand the computer and the system is much closer to 5 or 6 hours."

"I feel lost but I'm learning."

"Too much material to cover in the time allotted."

"Very interesting but I think I need more background in differential equations to fully appreciate the problems."

In the particular course being evaluated, some students had completed a course in differential equations, but most were taking differential equations concurrently. In talking to students who felt they were not receiving much benefit from the simulation terminals, it turns out that they were a semester behind in their mathematics sequence and would not take differential equations until next semester. Since the terminals were successfully used in the freshman orientation course, it is apparent that we used the wrong handout material for these students.

CONCLUSIONS

Student response to the use of this hybrid simulation laboratory has been very encouraging. It is apparent that we can indeed introduce the concepts of simulation

in the study of dynamic systems to all of our children. The key to success is the development of the student handout material which must be oriented toward both the specific course material and the background of the student. The cost of the system described is approximately \$80,000, not including the IBM 1130. The system can easily accommodate several thousand students per year. If the cost were amortized over a five-year period, the cost would be something on the order of eight to ten dollars per student per year for essentially unlimited use. Quite a bargain for simulation terminals when connect time, line charges and CPU time are considered for conventional digital terminals. While these terminals are more limited in scope, we are convinced that we will see significant improvement in the students' understanding of dynamic systems as the terminals are used in additional curricula.

ACKNOWLEDGMENT

The financial support of the National Science Foundation to develop and evaluate these hybrid terminals is gratefully acknowledged.

REFERENCES

- 1 G A KORN
The handwriting on the CRT
SIMULATION pg. 319 June 1969
- 2 D C MARTIN
A different approach to classroom computer use
ACEUG TRANSACTIONS Vol 1 No 1 January 1969
- 3 D C MARTIN
Development of analog/hybrid terminals for teaching system dynamics
Fall Joint Computer Conference 1970

BIOMOD—An interactive computer graphics system for modeling*

by G. F. GRONER, R. L. CLARK, R. A. BERMAN, and E. C. DeLAND

The Rand Corporation
Santa Monica, California

INTRODUCTION

Many of those involved in improving the quality of life often model and simulate continuous systems as part of their work. For example, one group of investigators may model an oil refinery to learn how to produce a new fuel efficiently, while another may simulate a global weather model to determine the effect of burning large quantities of the fuel at high altitudes. An urban planning team may simulate the water flow in an estuary to discover the best location for a new sewage treatment plant or the effect of a proposed breakwater. A medical team may simulate the bodily distribution of drugs to determine optimal dosage amounts and intervals, while another team might simulate the blood-volume control system to design a more efficient artificial kidney.

All of these investigators may take a common approach to solving their problem. This approach involves the following steps:

- Develop a mathematical model based on data and experience.
- Represent the model in terms suitable for an analog, hybrid, or digital computer.
- Run a computer simulation for a number of situations where the real system behavior is known.
- Adjust the model structure and parameters until it behaves the same as the real system.
- Run the simulation to predict the system behavior in new situations.
- Continue to collect data and run the simulation to verify the model and learn more about the real system.

* This research was supported by Public Health Service Grant No. 1-R01-GM-15896. Any views or conclusions contained in this paper should not be interpreted as representing the official opinion or policy of The National Institutes of Health or The Rand Corporation.

To accomplish this, the investigators must have available to them a computer system that can simulate large models and produce accurate, repeatable results. In addition, the computer system should provide aids for describing, analyzing, and verifying the model, mechanisms for changing the model and rerunning the simulation, and facilities for saving the model description, simulation run results, and data about the system being modeled.

An investigator can be most effective if he, the person who understands the real system, can directly and easily develop and operate the computer simulation of that system. The computer system should allow him to describe his model using terminology that is meaningful to him, to develop, modify and run his model by taking actions natural to him, and to examine his model's behavior in those ways that are most understandable to him. Because continuous systems are usually described by combinations of block diagrams, mathematical statements, logical operations, ordinary and partial differential equations, chemical equations, transfer functions, graphs, and tables of data, the computer system should be able to interpret these modes of representation. Modelers communicate by sketching diagrams, writing equations, drawing curves, and typing text, so the computer system should allow for these natural forms of input. Because the behavior of real systems is usually presented through graphs or through tables of numbers, the computer system should present the behavior of simulated models in these forms.

A succession of digital computer programs for simulating continuous systems¹⁻³ provides analog-computer-like elements, mathematical statements, logical operations, printed graphs, and the ability to automatically modify parameters and rerun. These programs are usually run in the batch mode, however, so an investigator can neither directly develop his model nor control its operation. Some recent computer



Figure 1—A BIOMOD user at a console

systems⁴⁻⁷ employ graphic consoles attached to computers to allow investigators to interactively construct and operate models. These systems are, however, still lacking in man-machine communication techniques. The BIOMOD system has been developed specifically to make modeling continuous systems more convenient for investigators who are unsophisticated in the use of computers. It accomplishes this by providing a high degree of interaction, graphical displays, user-oriented model-definition languages, and flexible, in-depth model structuring.

A modeler uses BIOMOD via an interactive graphics console comprising a television screen, data tablet, and keyboard (Figure 1). The system takes full advantage of the screen/tablet two-dimensionality by allowing the user to shift his attention from one place to another and take any appropriate action at will. He may communicate with the system by typing with the keyboard, and by handprinting or pointing with the tablet pen. BIOMOD responds immediately with its interpretation of user actions. This interaction is present in the system at several levels. For example, when the user handprints a character, the system displays its interpretation as a stylized character in the respective position on the CRT; when he completes a statement, the system either lists the variables appearing in the statement, or provides an error diagnostic. When he completes the description of his model, the system generates an executable program and runs the simulation.

A user may represent a model by a block diagram, each component of which, in turn, may be defined by another block diagram. This hierarchical structuring enables him to organize his model into meaningful substructures. At the most detailed level of the structure, he defines blocks by analog-computer-like elements, algebraic, differential, or chemical equations, and/or Fortran statements. A modeler may thus define his model in whatever terminology is meaningful to him.

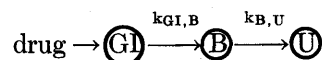
Displayed curves are continuously and automatically updated during model simulation. At any time, a user may stop the simulation, display curves for different variables, change scales, alter simulation parameters, or modify the integration method, and then continue the simulation or revise the description of his model.

The next section demonstrates BIOMOD by presenting a scenario of how a user might describe and simulate a simple, but important, model. The third section describes BIOMOD's features more fully. The paper concludes with a brief description of the BIOMOD system implementation and a discussion of experience with users.

A MODEL FOR EVALUATING DRUG ADMINISTRATION POLICIES

Medications and their prescribed dosages are designed to maintain a critical amount of drug in the blood for a specified period of time. The conventional method for determining optimal dosages involves numerous laboratory experiments. If the drug effects can be modeled, however, a more efficient method is to experiment by running computer simulations.

One technique for maintaining the prescribed amount of drug is to use a capsule containing a large number of differently coated pellets. The pellets dissolve at different times, so that, as drug leaves the blood, it is replaced by drug released by newly dissolved pellets. Garrett and Lambert⁸ have proposed the following model to describe this situation. A capsule comprises a number of pellet populations with different mean times of release. The rate of drug release for each population is assumed to be normally distributed (with the same standard deviation for each population) about the mean time of release for that population. The rate of adding drug to the body is therefore specified by a sum of normal distributions. The transfer of drug through the body is described by



where GI refers to the gastrointestinal tract, B to the

blood, and U to the urine and other excretory parts of the body; this means that drug flows from GI to B at rate $k_{GI,B}$ and from B to U at rate $k_{B,U}$. Thus, for example,

$$dD_B/dt = k_{GI,B}D_{GI} - k_{B,U}D_B$$

where D_B is the amount of drug in B, and D_{GI} the amount of drug in GI.

Given this description together with the requisite parameter values, BIOMOD can be used to simulate the model. Note that the following dialogue describes only one of many possible ways of reaching the same goal, and that BIOMOD does not force the user to take actions in any particular order.

When using BIOMOD, we communicate via a data tablet pen and a keyboard. The pen's location on the tablet is always indicated by a dot displayed in the corresponding location on the television screen. BIOMOD's interpretation of user pen actions depends on where the pen is placed and on what is currently displayed on the screen. We may handprint characters in most areas. As we write, a displayed "ink" track appears to flow from the pen; each time we complete a character, its track is replaced by a stylized character. We can change a character by writing another over it. Some symbols are used for editing; for example, we may use a caret to insert text, or we may scrub with the pen to delete text. Some areas displayed on the screen act as pushbuttons; if we "push" one of these (by touching the pen down), the system performs the indicated action. If we push a displayed arrow, a continuous action takes place, such as the rescaling of a set of curves. Some figures can be "dragged"; if we "touch" one of these and move the pen, the displayed figure follows the pen's motion. We may type (with the keyboard) in any area where writing is possible. The keyboard cursor may be positioned either with the pen or with keyboard control keys.

To create our model, we enter our identification, name our model DRUGS, then begin constructing the model. Because it has two major components, we first draw two rectangles; these are replaced by stylized function boxes. We write CAPSULE in one box and names of parts of the body in the other box, and then draw a flowline to connect them. This diagram (Figure 2) provides not only a picture of our model, but also a means of defining the two components of the model separately.

To define the capsule component, we first push the DEFN button on its box. The system replaces the block diagram with a list of languages that we may choose from to define the component. The languages are: block diagrams, mathematical equations, chemical equations, and Fortran statements. We choose block diagrams so

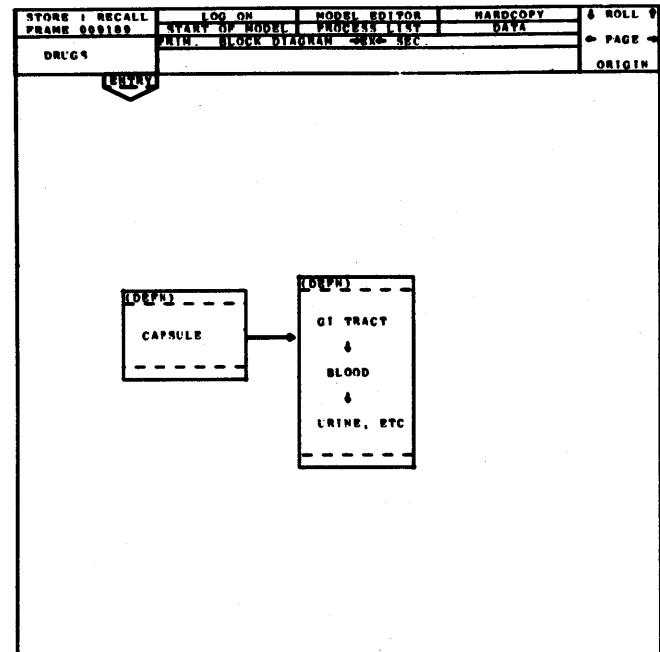


Figure 2—The DRUGS model block diagram

that we can define the capsule as a set of boxes, each representing a pellet population. This ability to define a box by another block diagram enables us to organize a model as a hierarchical collection of a number of components at different levels. We draw four boxes and write PILL on the top line of each to represent four pellet populations. We name a function box in this way whenever we anticipate using the same function repeatedly.

Because each population is defined by a normal distribution, we indicate that we want to define the PILL function with mathematical equations. BIOMOD responds by displaying a form for writing algebraic and differential equations. We assume that the probability of a pellet dissolving in an interval about time t is given by the probability density function

$$p = \frac{1}{\sigma\sqrt{2\pi}} \exp[-(t - m)^2/2\sigma^2]$$

Using this function to approximate the drug release rate by a deterministic variable, we write

$$P = 1/(\text{SIGMA}*\text{SQRT}(2*\text{PI}))*\text{EXP}(-(\text{TIME}-\text{MEAN})**2/(2*\text{SIGMA}**2))$$

The system analyzes this statement and immediately responds with the message

UNBALANCED PARENTHESES

STORE RECALL	LOG ON	MODEL EDITOR	HARDCOPY	ROLL
FRAME 000000	START OF MODEL	PROCESS LIST	DATA	PAGE
PILL	WITH MATH EQUATIONS	SEXC SEC	DATA	ORIGIN
ENTRY				SCROLL

MATHEMATICAL EQUATIONS CODING FORM

DEFINES	DEFINITION
P	P = DOSAGE / (SIGMA * SORT(2 * PI)) * EXP(-(TIME - MEAN) ** 2 / (2 * SIGMA ** 2))

VARIABLES DEFINED HERE		VARIABLES DEFINED ELSEWHERE			
CHECK IF: OUTPUT VARIABLE LOCAL VARIABLE	NAME	COMMENT	CHECK IF: RENEWABLE INPUT NON-RENEWABLE INPUT	NAME	COMMENT
	P			MEAN	
				PI	
				SIGMA	
				DOSAGE	

Figure 3—The definition of a pellet population

We then add a closing parenthesis to correct the statement. We also realize that we should parameterize the amount of drug released by each pellet population, so we insert DOSAGE after the equals sign, and scrub the 1. The display now appears as in Figure 3.

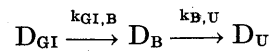
BIOMOD has generated separate lists of the defined and undefined variables; TIME does not appear because it is always the simulation independent variable. These lists enable us to indicate which variables have different meanings or values each time we use the function. We indicate that the names (and therefore the values) of PI, SIGMA, and DOSAGE are the same each time we use the PILL function. This is because PI is a constant, and because we assume that the standard deviation and dosage amount are the same for each population. On the other hand, we indicate that MEAN may have a different value for each pellet population.

Now that we have defined the PILL function, we are ready to use it to define the individual populations. We push a button to get back to our diagram of the four PILL boxes, then push the DEFN button on one of these. Because PILL is now defined, BIOMOD displays

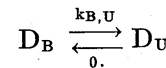
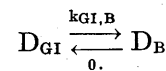
P →
 MEAN ←
 PI ← PI
 SIGMA ← SIGMA
 DOSAGE ← DOSAGE

for us to provide the names of the output and mean of this particular population. We write P1 next to P →, to name this output P1, and write M1 next to MEAN ←, to name this mean time of release M1. We similarly establish the correspondence between the names of variables in the other three pellet populations and the names (P and MEAN) used when defining the function PILL.

We can describe the flow of the drug through the body by chemical equations because these are mathematically equivalent to mass transport equations. When we push the DEFN button on the box that describes the body, and select chemical equations, BIOMOD presents an appropriate form. According to our original model description, we would like to write



or



where 0. indicates that there is no backward flow. BIOMOD requires that we linearize each equation, and write the rate coefficients and equation in the provided columns as

S	KGIB	0.	DGI = DB
S	KBU	0.	DB = DU

Here S (for slow reaction) means that BIOMOD should derive integral equations from our equation. Since the pellets release drug into the gastrointestinal tract, we also write

$$G \quad DGI \quad P1 + P2 + P3 + P4$$

This statement (with G for gain) indicates that the gain of DGI, i.e., the increased rate of change of DGI due to drug entering the body from outside, is equal to the sum of the rates of drug release from the four pellet populations.

The model is now defined except for parameter values. When we indicate that we are ready to provide these values, BIOMOD displays the names of model variables and parameters in two separate lists (Figure 4). Names such as DGI0 indicate initial values; they are derived from the chemical equations by BIOMOD. We enter the values given or implied by Garrett and Lambert. We assume that some drug is immediately released into the gastrointestinal tract and therefore set DGI0 to 5; the other drug amounts are initially

STORE RECALL FRAME 000100	LOG ON START OF MODEL DATE	MODEL EDITOR PROCESS LIST DATE	HARDCOPY DATA	ROLL ↑
DRUGS	DATE	DATE	DATE	PAGE
				ORIGIN

VARIABLES		PARAMETERS	
NAME	COMMENT	NAME	VALUE COMMENT
DGI		DU	0.0000E+00
DB		RSU	1.0000E+01
DU		DG1	0.0000E+01
PI		DG2	0.0000E+01
		RSI	0.0000E+01
		NS	0.0000E+01
		NS4	0.0000E+01
		NS2	0.0000E+01
		NS1	0.0000E+01
		PI	0.0000E+01
		SIGMA	0.0000E+01
		DOSAGE	0.0000E+01

Figure 4—The model variables and parameters

zero. In order to minimize storage requirements, BIOMOD limits the number of variables whose values are saved during simulation and the number of parameters whose values can be modified during simulation. Since this model is small, we indicate that we want to save values of (and possibly plot) all the variables, and that we might want to modify values of all parameters except PI.

It has taken us less than half an hour to completely describe our model. We now indicate that we would like to simulate it. BIOMOD first produces a CSMP/360² program that describes our model and provides for graphic display of the results. CSMP, in turn, generates a Fortran program, which is compiled and linked with other programs required to run the simulation. If an error is detected at one of these steps, program listings and error messages are displayed on the screen; otherwise, our only awareness of the intermediate steps is via displayed messages. The time required for the translations depends on the load on the (multiprogrammed) computer; generally it is about three minutes.

Our DRUGS model translates successfully so the form shown in Figure 5 is displayed on the screen. As in the other forms, software pushbuttons appear across the top. The central area is for selecting numerical integration methods, modifying parameter values, examining variable names and values, or plotting

graphs. The areas to the left and below the central area are for specifying the y and x axes of the graphs.

We expect the values of our model variables to change smoothly and over several units to TIME, so we choose a simple integration method—Simpson’s method with step-size = 0.1. This is a fixed step-size method, so the information regarding variable step-sizes disappears. Before studying how to use a multi-pellet capsule, we want to ensure proper model behavior when there is initially some drug in the gastrointestinal tract, but no capsule. To eliminate the capsule drug we push the PARAMETERS button to display the list of modifiable parameters in the central area, then overwrite the value of DOSAGE, changing it to 0. Next we display the list of plottable variables. Because we are most interested in the amount of drug in the gastrointestinal tract, blood, and urine, we drag the names DGI, DB, and DU to the y axis. We want to watch the model for several simulated hours, so we change the upper range of TIME (in the small box at the lower right of the central area) from 1. to 7. We push PLOT; now we are ready to plot DGI, DB, and DU from 0. to 1. against TIME from 0. to 7. hours.

We push RESTART and the simulation begins running. We see (from the curves) that DB, and later DU, are being generated; the “NOW X =” number changes continuously to indicate the current value of simulated TIME. Because DGI is plotted off scale

FOR CURVES:	DO ↑	DISPLAY ↑	GO TO ↑
ENTER	RESTART	CONTINUE	PARAMETERS
ANALYZE	STOP	HARDCOPY	TEMPORARY COPY
PLOT	THIN ALL DATA	THIN THIS DATA	VARIABLES
INTEGRATION	FOR RUN 1, DRUGS	J METHODS	MODEL EDITOR
			START OF MODEL
			08/23/71 14:02

Y-NAME	RUN	ID

1.00E+00	INTEGRATION METHOD
LINEAR	() RECTANGULAR
	() TRAPEZOIDAL
	() SIMPSON'S RULE
	() 2ND-ORDER ADAMS
	() 4TH-ORDER RUNGE-KUTTA
	(X) 4TH-ORDER RUNGE-KUTTA, VARIABLE STEP-SIZE
	() 5TH-ORDER MILNE, VARIABLE STEP-SIZE
	() FOWLER-WARTEN MOD B AND C, VAR. STEP-SIZE
	INITIAL INTEGRATION STEP-SIZE = 1.00E-08
	MIN ALLOWABLE INTEGRATION STEP-SIZE = 1.00E-10
	MAX REL ERROR IN INTEGRATOR OUTPUTS = 1.00E-04
	MAX ABS ERROR IN INTEGRATOR OUTPUTS = 1.00E-03
	THE STEP-SIZE IS ADJUSTED SUCH THAT:
	ESTIMATED ERROR
	A+R*ABS(Y) ≤ 1
	WHERE Y = ESTIMATED OUTPUT
	R = SPECIFIED MAX RELATIVE ERROR
	A = SPECIFIED MAX ABSOLUTE ERROR
0.00E+00	LINEAR
0.00E+00	X-NAME = TIME
1.00E+00	NOW X = 0.00E+00
	PLOTTING INTERVAL = 0.00E+00
	STOP IF X > 1.00E+00

Figure 5—The simulation control form with integration methods

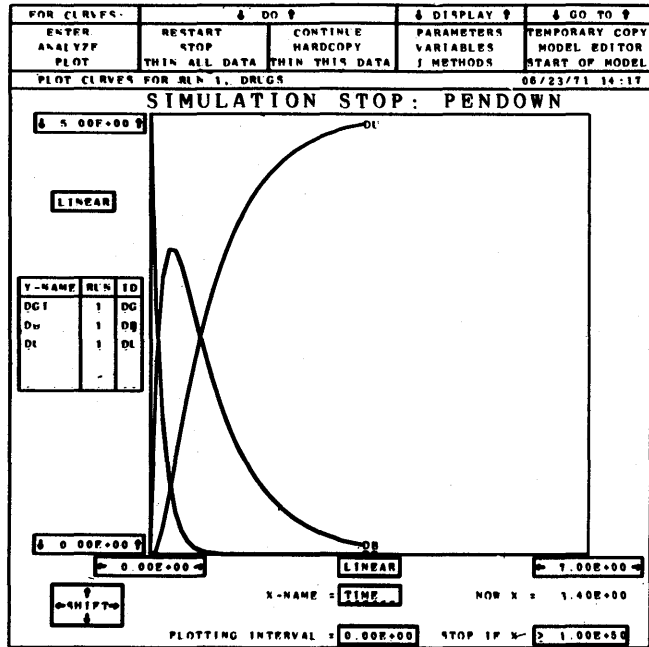


Figure 6—The simulation run with no capsule drug

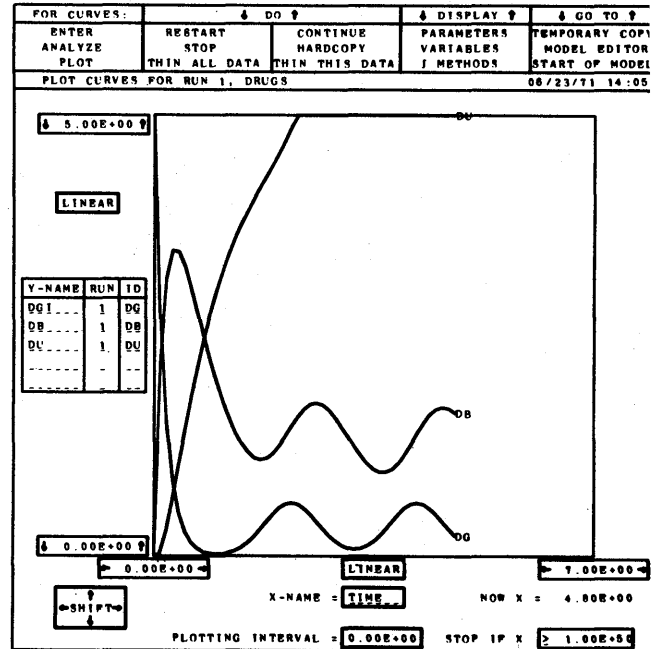


Figure 7—The simulation run with the parameter values shown in Figure 4

along the upper boundary, we touch the pen down to stop the simulation. In order to determine the range of DGI, we return to the display that lists the variables along with their current, minimum, and maximum values. The maximum value of DGI is 5. (its initial value), so we write 5 over the 1 that specifies the upper y-axis value, then redisplay the curves. The curves are now nicely scaled. We continue the simulation, then stop it when we see that nearly all the drug has entered the urine, and values are changing slowly. We assume, from the curve's reasonable appearance (Figure 6), that we described at least the body component of the model correctly, and our choice of integration method is adequate. We see from the curve labeled DB that, as expected, the drug remains in the blood for only a short time.

We reintroduce the capsule drug by changing DOSAGE back to 3., then restart the simulation and watch the curves being continuously updated as it runs. Once it becomes apparent that the capsule is not effective, i.e., that the value of DB drops too low, we stop the simulation. Apparently (Figure 7) the drug is not released from the pellets in time to replace the drug that leaves the blood. To correct this, we change the mean times of release from 2., 4., 6., and 8. to 1., 2., 3., and 4. We then rerun the simulation and get much better results. The amount of drug in the blood should

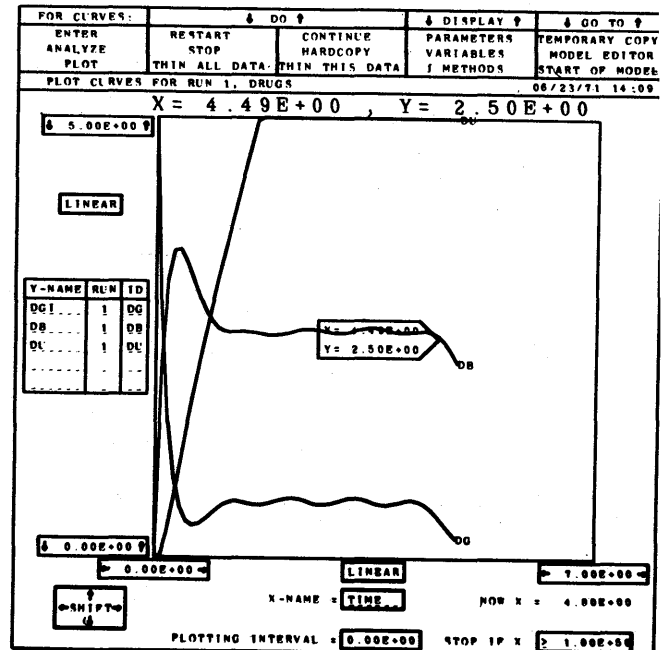


Figure 8—The simulation run with Means = 1, 2, 3, and 4.

be greater than 2.5. In order to determine if this is achieved, we place the pen down in the central area to establish an x-y meter, then drag this meter to a place where Y (corresponding to DB as well as DGI and DU) is equal to 2.5 (Figure 8). Our choice of means was good, but they need to be adjusted to maximize the total duration of capsule effectiveness.

While changing the means for further trials we realize that rather than controlling four means, we would prefer to deal only with the first mean and the interval between mean times of release. To reformulate the model in this way, we return to its description and add another box to the definition of the capsule component. In this box we write

$$M2 = M1 + INTVAL$$

and similar equations for M3 and M4. This replaces the parameters M2, M3, and M4 with the single parameter INTVAL, which we set to 1. and mark modifiable. Once that is accomplished, we retranslate the model, then continue to resimulate it and change parameters until we have established a satisfactory drug formulation and administration policy.

SOME ADDITIONAL BIOMOD FEATURES

The DRUGS example illustrates many, but not all, of BIOMOD's features. One facility that was not described is file management. Every new model is saved on secondary storage, and is filed according to name and user identification until it is intentionally destroyed. One may copy, and then modify a model description to build a family of related but different models.

When drawing a block diagram or writing a set of equations, a user may run out of space on a displayed "page." In either case, a blank continuation page may be obtained by pushing a displayed button. A few lines of text may similarly be moved off the display to create writing space.

Text can be edited by overwriting, deleting, closing, inserting between characters, and inserting between lines. Block diagrams can also be conveniently edited. A box or a flowline may be deleted by scrubbing. The appearance of a block diagram may be improved by dragging a box to another position or by "stretching" it (from its lower right corner) to change its size and shape.

BIOMOD makes available most of the functions provided by CSMP/360. These include mathematical functions, logical functions, and signal sources. A box given the name of one of these functions has its defini-

tion provided by the system. Such a box is used in the same way as a user-defined function (e.g., PILL), except that its definition cannot be viewed. Equations may also refer to these functions; BIOMOD checks to see that such a reference includes the proper number of arguments.

Mathematical equations may be differential equations as well as algebraic equations. Derivatives with respect to time are indicated by up to nine prime signs (') or by a prime sign and a digit. The variable defined by an equation need not appear alone to the left of an equals sign. Thus, the equations

$$\begin{aligned} M_1\dot{x}_1 + (K_1+K_2)x_1 - K_2x_2 &= 0 \\ -K_2x_1 + M_2\dot{x}_2 + B\dot{x}_2 + K_2x_2 &= 0 \end{aligned}$$

may be entered as

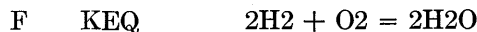
$$\begin{aligned} M1*X1'' + (K1+K2)*X1 - K2*X2 &= 0. \\ -K2*X1 + M2*X2'' + B*X2' + K2*X2 &= 0. \end{aligned}$$

If the user indicates that the first equation defines X1, BIOMOD manipulates it to place X1'' alone at the left, generates integral equations for X1' and X1, and requests initial values for X1 and X1'. The second equation is handled similarly. The major restrictions are that the highest derivative of the defined variable may appear only once in an equation, and may not appear as a function argument.

Chemical equations may be written as



or as



In the former case, S indicates that BIOMOD is to generate, and numerically solve, integral equations that model the (slow) chemical reaction; rate coefficients (KF and KB in this example) indicate the rate at which the reaction proceeds in each direction. In the latter case, F indicates that the (fast) reaction is to be forced to equilibrium initially and at each successive time step; an equilibrium coefficient (e.g., KEQ) is used in specifying the equilibrium condition. In either case, BIOMOD requests initial values of the chemicals. Rate and equilibrium coefficients may be defined by either algebraic expressions or numerical values on a chemical equations form. Each box defined by chemical equations is treated as a self-contained compartment separated from the others. Mass flow between compartments is specified by gain terms as in the DRUGS example. A non-reacting chemical may be

included in a compartment to affect the concentrations of the other chemicals.

A function that involves the conditional evaluation of variables may be specified entirely in terms of Fortran statements. The allowable statement types are assignment, arithmetic IF, GO TO, and CONTINUE.

While running the simulation, any set of up to five variables may be plotted against TIME or any other variable. The plots may be linear, logarithmic, or semi-logarithmic. The ranges of the axes may be changed by overwriting the numbers that specify them, dragging variable values over these numbers, pushing displayed buttons to gradually magnify or contract the curves, or pushing buttons to shift the curves. When the curves are rescaled or shifted, they are re-displayed so fast that they appear to magnify or move continuously. As the simulation proceeds, the user may control the intervals at which points along the curves are plotted, and those at which data values are saved.

Once he begins simulating a model, a user may like to define a new variable for some purpose, e.g., to scale a variable so that it has approximately the same range as others or to plot a boundary value. BIOMOD allows a user to define such a new variable as a simple combination of constants and other variables without requiring retranslation of the model. Such a new variable may be plotted just as any other. It is not incorporated into the model, however, and so it cannot define a parameter such as M2 in the DRUGS model.

If a user wishes to reexamine his model description during the simulation process, but not modify the model structure, he indicates this by pushing the TEMPORARY COPY button. BIOMOD saves a translated copy of the most recently simulated model so that, after examining the model, a user may simulate it again without retranslation.

A user may push a displayed button to request hardcopy of what is currently displayed. Hardcopy is produced off-line on a Stromberg Datagraphix 4060 film and hardcopy unit. The figures in this paper (except Figure 1) were generated this way.

IMPLEMENTATION

The BIOMOD system operates on an IBM System/360, Model 40 or larger, utilizing a partition of approximately 228,000 bytes. The operating system may be either the MFT II or MVT version of OS/360, augmented by Rand's Video Operating System, which serves as a link to the Rand Video Graphic System.⁹ BIOMOD is used from a Video Graphic console comprising a television screen, a data tablet, and a keyboard.

Parts of the model description portion of BIOMOD were derived from GRAIL,¹⁰⁻¹² a Rand system that enabled users to draw and execute program flowcharts. Both BIOMOD and GRAIL were tailored to provide good response times for operations on complex problem descriptions while minimizing demands on computer resources. However, since they are designed for different applications, they differ in their user-oriented languages.

When a user requests simulation of his model, the BIOMOD translator interrogates the data structure that contains the model description, and produces a CSMP program. Dummy names are generated for variable names that include primes (') or initial value symbols (o). Algebraic and differential equations are rearranged so that a single variable is assigned a value specified by an expression. Chemical equations are handled by calls to specially designed subroutines. Boxes defined by Fortran statements are implemented as CSMP procedures. User-defined functions are implemented as CSMP macros. In addition to this description of a model, the BIOMOD-generated CSMP program also includes calls to graphics subroutines that let the user control the simulation and observe results. The program also includes a subroutine that communicates the names and storage addresses of variables and parameters to the graphics subroutines.

The CSMP program is passed to the standard CSMP/360 processor. CSMP sorts statements and expands macros to produce a Fortran program that describes the model, and then calls the Fortran compiler and the Linkage Editor to generate an executable program. The resulting program runs under CSMP control. The simulation graphics programs are Fortran subroutines that call assembly language routines to extend the capabilities of Fortran and to communicate with the graphics hardware. They extract variable values and change parameter values directly; they return codes to control the simulation.

Since a user's model is represented at one point in the translation as a CSMP program, a user with a batch-mode CSMP program may modify it slightly, then load it into the BIOMOD system to take advantage of the simulation graphics facilities. Similarly, a user may modify a CSMP program produced by BIOMOD, then run it at another facility in batch mode.

CONCLUSIONS

Users are very enthusiastic about BIOMOD. The combination of graphics, highly interactive facilities, and user-oriented languages enables them to get their work done quickly, and occasionally provides insights

that would have been missed using other techniques. The graphics not only help in visualizing the model and the simulation results, but also provide for operating freely on a two-dimensional surface.

For the most part, the interactive techniques have been well received. The methods for controlling the simulation and manipulating graphs are particularly effective. The pen and tablet are intended to be used like pencil and paper; however, users tend to use the tablet pen for printing labels, changing values, editing text, and dragging, but use the keyboard for entering equations and anything else that is extensive, because typing is faster than printing.

Thus far, most BIOMOD users have had previous experience writing and running batch-mode programs, and this has influenced the way they develop models. They tend to be too experienced as programmers to require all the aids provided by BIOMOD, yet too inexperienced as modelers to take advantage of all the power provided. For example, those of our users who have previously written Fortran and CSMP programs usually state their models in terms of these formal languages, rather than write differential and chemical equations. On the other hand, users with little or no computer experience find it convenient to describe models in their own terminology. Since it is often interesting to study models in the literature, it is particularly convenient to be able to directly transcribe their description with, perhaps, minor notational changes.

Block diagrams are used more to organize the model into component parts and to make different languages available for defining some boxes than they are for using system functions, defining new functions, or visualizing the flow of signals or mass. Our users have not learned to take advantage of BIOMOD's hierarchical capability, but rather, construct models at only one or two levels. This is probably because they have not yet begun to develop very complex models, and because hierarchical model structuring has previously been difficult to accomplish. Another reason is that the techniques for moving from one part of a model to another are presently too complicated, particularly when user-defined functions are involved.

BIOMOD has several inadequacies that we plan to correct. It is often convenient to specify the initial state of a model by a set of equations that are evaluated only once; BIOMOD does not provide for this. Users would like to draw data curves to define functions or to compare with simulation results. Although the tablet is an excellent device for drawing, this feature is not yet implemented. Users would like to save simulation results in order to compare various runs, but they can-

not. Users spend a great deal of time adjusting parameter values and rerunning the simulation until they get the desired results; it would be very helpful to use parameter identification schemes to automate this process. Most of these features were taken into consideration when BIOMOD was initially designed and can be added without any major difficulty.

Other problems are hard to remedy. Users with large models run into size limitations imposed both by BIOMOD and by CSMP. These limitations can undoubtedly be relaxed, but it will require more experience to evaluate tradeoffs, and may require abandoning the standard version of CSMP. Many CSMP and execution-time error messages are vague. In the present implementation it is difficult, if not impossible, to automatically relate these to a specific part of the user's model. BIOMOD operates only at The Rand Corporation because of its display-hardware dependencies; we are presently rewriting large portions of the system to make it exportable.

Further details about the BIOMOD system are reported in a user's manual¹³ and in a description of its implementation.¹⁴

REFERENCES

- 1 J J CLANCY M S FINEBERG
Digital simulation languages: A critique and a guide
AFIPS Conference Proceedings 1965 FJCC Vol 27 pp 23-36
Spartan Books Washington D C 1965
- 2 *System/360 continuous system modeling program (360A-CX-16X) application description*
IBM Corporation Form No H20-0240-2 August 1968
- 3 SCI SIMULATION SOFTWARE COMMITTEE
The SCi continuous system simulation language (CSSL)
Simulation Vol 9 No 6 pp 281-303 December 1967
- 4 H B BASKIN S P MORSE
A multilevel modeling structure for interactive graphic design
IBM Systems Journal Vol 7 Nos 3 and 4 pp 218-229 1968
- 5 R G RENAUD R F WALTERS
The interactive creation, execution and analysis of biological simulation using MIMIC on a graphic terminal
Proceedings of the Conference on Applications of Continuous System Simulation Languages San Francisco California pp 185-191 1969
- 6 G A KORN
Project DARE: Differential analyzer replacement by on-line digital simulation
AFIPS Conference Proceedings 1969 FJCC Vol 35 pp 247-254
AFIPS Press Montvale New Jersey 1969
- 7 M J MERRITT D S MILLER
MOBSSL-UAF—An augmented block structure continuous system simulation language for digital and hybrid computers
AFIPS Conference Proceedings 1969 FJCC Vol 35 pp 255-274
AFIPS Press Montvale New Jersey 1969

- 8 E R GARRETT H J LAMBERT
Analog computer in drug dosage and formulation design
Journal of Pharmaceutical Sciences Vol 55 No 6
pp 626-634 June 1966
- 9 K W UNCAPHER
*The Rand video graphic system—An approach to a general
user-computer graphic communication system*
The Rand Corporation R-753-ARPA April 1971
- 10 T O ELLIS J F HEAFNER W L SIBLEY
*The GRAIL project: An experiment in man-machine
communication*
Proceedings of the Society for Information Display
Vol 11 No 3 pp 121-129 Third Quarter 1970
- Also The Rand Corporation RM-5999-ARPA
September 1969
- 11 T O ELLIS J F HEAFNER W L SIBLEY
The GRAIL language and operations
The Rand Corporation RM-6001-ARPA September 1969
- 12 T O ELLIS J F HEAFNER W L SIBLEY
The GRAIL system implementation
The Rand Corporation RM-6002-ARPA September 1969
- 13 R L CLARK G F GRONER R A BERMAN
The BIOMOD user's reference manual
The Rand Corporation R-746-NIH July 1971
- 14 R L CLARK G F GRONER
The BIOMOD system implementation
The Rand Corporation R-747-NIH July 1971

The future on-line continuous-system simulation

by HANS M. AUS and GRANINO A. KORN

The University of Arizona
Tucson, Arizona

INTRODUCTION AND REVIEW

The DARE I and DARE II simulation systems each added a simulation console with graphic and alphanumeric displays to a PDP-9 minicomputer with 16K of memory and a small disk (Figure 1) and employed a continuous-system simulation language for simplified programming. System equations or block statements, text, and comments, are typed and edited on a CRT typewriter. Solutions appear on a second CRT and can be plotted or listed for report preparation; they are automatically labeled and scaled without any need for special FORMAT statements. Iterative and statistical simulation studies involving repeated differential-equation-solving runs are possible.

A DARE system is loaded onto the small PDP-9 disk from a reel of magnetic tape. The TYPE EQUATIONS console light lights, and a "communication line" at the bottom of the alphanumeric CRT says

DERIVATIVE BLOCK NO. 1—INPUT MODE

The operator types first-order differential equations, say

$$X' = XDOT$$

$$XDOT' = ALFA*(1. - X*X)*XDOT - X$$

and equations introducing "defined variables," such as

$$E = X - BETA*SIN(X)$$

in any order. He can intersperse this material with titles, comments, and other report material; each such "comment line" begins with a star to prevent compilation. Program and text can be edited at will on a CRT typewriter, which permits one to move words or lines, to substitute symbols, and also to find specified symbol strings in long programs. Any or all of this material can also be printed out as a hard-copy report at the touch of a console button.

Table look-up functions of one or two variables are simply entered as one- or two-dimensional tables called

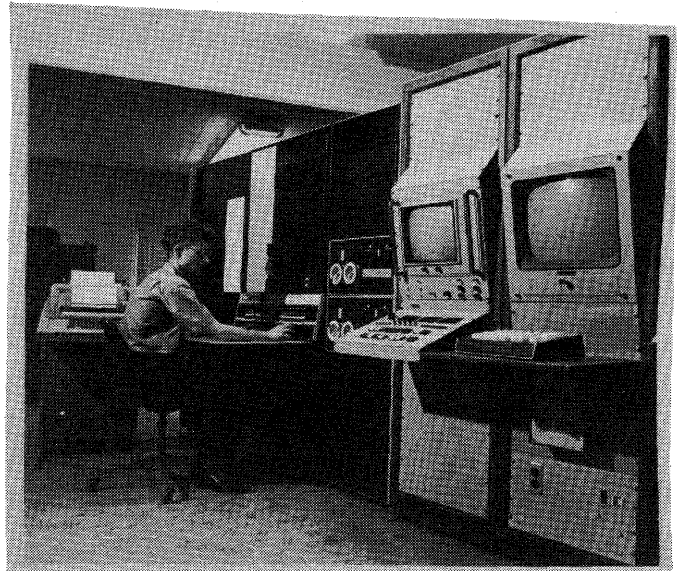


Figure 1—PDP-9 and DARE console at the University of Arizona

by function names. An initial display, showing one or two variables against the independent variable T, or a phase-plane plot, is specified with a display statement, say

DISPLAY X, XDOT, T

Note, however, that *all* state variables and defined variables are also stored for later display or listing in any combination.

The integration routine to be used is selected with a 12-position console switch (DARE I and II), or by typing the method number on the CRT screen (DARE III). There is a choice of predictor/corrector- and both fixed- and variable-step Runge-Kutta methods, plus an "implicit" method for stiff-equation systems (DARE I, DARE II, and DARE III have *two* derivative blocks, permitting simultaneous use of two different integration methods and step sizes).

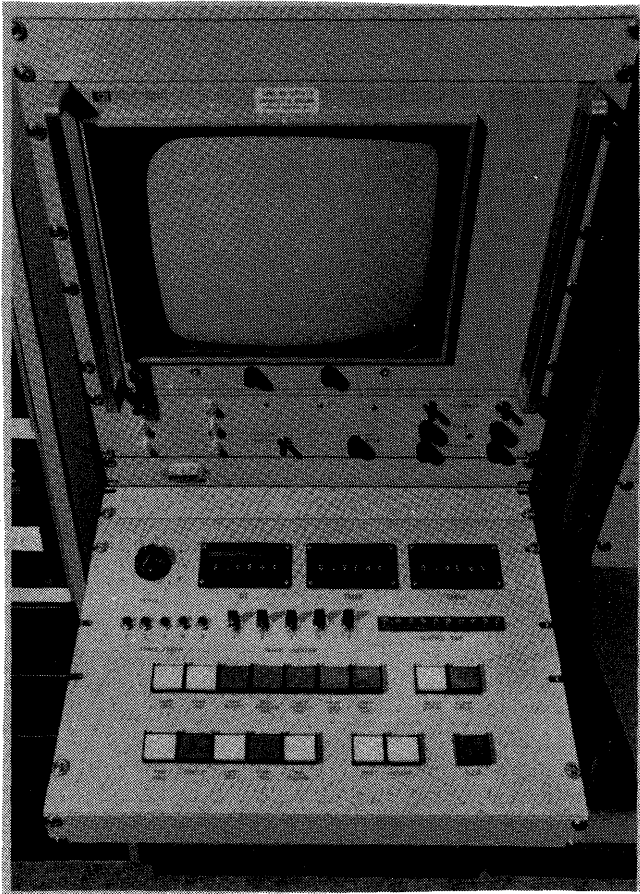


Figure 2—Closeup of DARE control panel, showing method switch, sense switches, and lighted control buttons

If we want to run a complete simulation study requiring multiple differential-equation solving runs, iterative adjustment of parameters or initial values, and/or crossplotting or statistical evaluation of results obtained in successive runs, we type OPEN LOGIC to call for a DARE logic block and proceed to type a FORTRAN IV program such as

```
CALL RUN
ALFA=2.5
CALL RUN
ALFA=3.7
etc.
```

This logic block will then take control of the computation and call for successive equation solving runs with suitable parameter changes, which could also depend on results from past solutions.

Our program is now complete except for initial-value and parameter settings (corresponding to potentiometer

settings on an analog computer. We push the COMPILE button on the console.

WHAT THE SYSTEM DOES: STAND-ALONE-COMPUTER SYSTEMS

The edited DARE I, DARE II, or DARE III program (source-language program) is partially in core and partially on the small local (PDP-9) disk. To see what each DARE system must do, let us first look at a stand-alone-minicomputer system, say DARE I, which can handle 20 first-order state equations. This will make it easier to understand the larger time-sharing systems.

The COMPILE button causes compilation and loading in several overlays from the minicomputer disk. A precompiler (translator) first sorts the system equations into a FORTRAN program so that no statement can call for as yet uncomputed quantities. Undefined parameters (BETA, and the first-run value of ALFA in our example) are left over in this sorting process and will be presented automatically to the operator, who must supply numerical values. The FORTRAN compiler is loaded next and compiles the FORTRAN program, including the logic block. Finally a linking loader loads the resulting binary program together with any library routines needed (such as sine or square-root functions). The alphanumeric CRT screen now displays the names of all initial-value settings and as yet undefined parameters, say,

```
X=          XDOT=
ALFA=       BETA=
```

together with the simulation parameters DT (integration step), TMAX (total computation time) and, in variable step integration routines, also EMAX, the maximum allowable local truncation error.

The operator then simply enters the desired values on the CRT typewriter. Note that he did not have to remember which quantities needed to be specified; the CRT screen told him.

We are now ready to solve the differential equations. Pushing the COMPUTE button on the console starts the computation; the initial solution display will appear on the graphic-display CRT. If there is a logic block, solutions will automatically proceed through the desired iteration sequence.

After a solution is complete, the operator can push a RESET button on the console to display the parameter values again, type new ones, and restart the solution at once by pushing the COMPUTE button (this corresponds to resetting and restarting an analog-computer solution). It is also possible to change the integration routine, DT, TMAX, and EMAX without

recompiling, and to use sense switches on the console. If one wishes to change the differential equations in a more radical way, one pushes the RESTART button on the console; this will again display the differential equations, which can now be changed and recompiled.

After a set of differential equations has been solved, the SELECT DISPLAY button on the console loads another PDP-9 overlay which permits one to recall time histories of all state variables and/or defined variables, and also crossplots from different solutions, from the disk. The following display options are obtained by simply typing codes on the alphanumeric CRT typewriter.

1. Plot up to 4 variables against T on the same or different reference axes. Curves can be in 4 colors.
2. Plot any variable against any other (phase-plane plots).
3. Tabulate up to 4 variables on CRT or teletypewriter.
4. Obtain hard-copy plots on the 4—channel strip-chart recorder or XY recorder.

The possibility of plotting any set of variables against time or any other variable, including variables saved from preceding computer runs by a special option code, is not only very convenient for evaluation of results and report preparation, but also constitutes a useful debugging aid.

TIME-SHARING SYSTEMS

Large simulation problems require more powerful computers, and such computers are too expensive to wait idly while an engineer engaged in on-line simulation thinks or interprets results. We cannot tie up a large computer for interactive simulation without some type of time-sharing. A medium-sized computer (of the order of an XDS SIGMA 5) with a sufficiently large sector-protected memory could be time-shared between one interactive simulation and a batch-processed background program, which would be interrupted by the simulation runs.

Perhaps the main attraction of such a system is that the simulation laboratory would have its own computer. For more cost-effective time-sharing of a larger digital computer, and for multiple interactive simulations, simulation programs will have to be swapped in and out from a system disk. It would also seem expedient to arrange priorities and time slots so that each simulation runs or each iterative sequence of simulation runs is considered as a job which ordinarily cannot be interrupted by other users. Typical runs might take seconds,

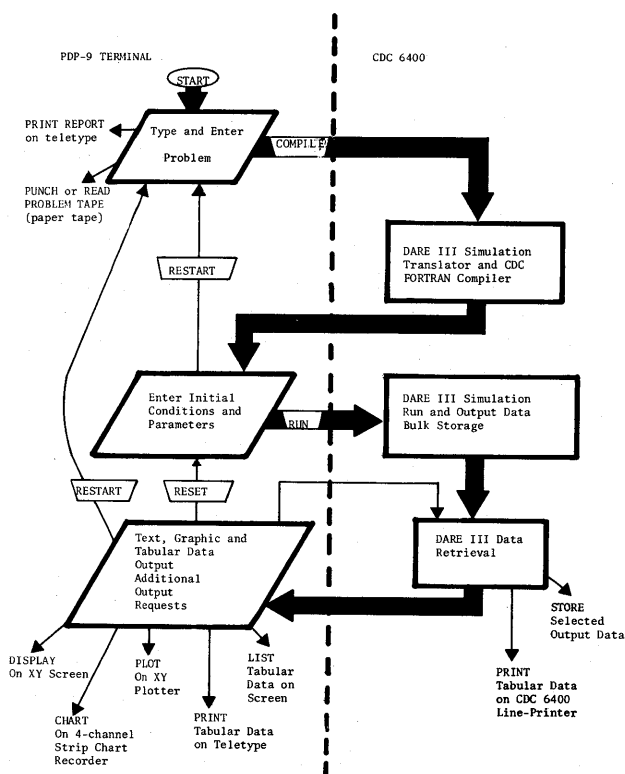


Figure 3—Time sequential flow chart of DARE III

but large iterative sequences could take much longer. Quite frequently, interactive simulation would be employed mainly to debug some initial runs, with the rest of the study deferred for batch-processing study at night.

In developing DARE III, we were confronted with the ponderous organization of a university computing center with a CDC-6400 in an iron-clad operating system (CDC SCOPE), which we did not want to change. The closest approximation to time-sharing was the CDC INTERCOM system used at the University mainly for remote batch-processing job entry and printing. Since the 6000-series INTERCOM combination is widely used in Universities, it was a worthwhile challenge to develop a suitable simulation system.

DARE III OPERATION

Most of the user interaction required in the DARE III system is the same as in DARE I. The main difference between the interactive portion of the two systems is that DARE III does not use the simulation control panel, and that the user must dial, connect and disconnect the telephone as directed from the CRT screen. The DARE III user can exercise system control functions exactly like those provided by the control

panel in DARE I by typing the appropriate command on the last line of the CRT screen, for example COMPILE, PRINT REPORT, READ PROBLEM TAPE, etc.

The simulation-problem text is entered and modified using the alphanumeric CRT editor. Each problem-definition block can contain up to 600 full 40 character lines. DARE III also offers sophisticated users the ability to replace any run-time system routines with his own routines such as new integration routines, transfer-function operators, etc. These may be written in either CDC FORTRAN IV or 6000 series assembly language (COMPASS).

After the user has entered and modified his problem, he types COMPILE on the last line on the CRT screen. The screen next flashes

DIAL COMPUTER 3243

Using the Data Phone adjacent to the keyboard the user dials the university extension 3243 and presses the DATA button after the initial answer-back tone is finished. Several Control Data INTERCOM messages flash up on the screen for user information. The user does not take action in response to any message except

HANG UP THE PHONE

When the telephone has been disconnected, the screen will say:

PLEASE SELECT INTEGRATION METHOD

After the user has selected a valid integration-routine number, the alphanumeric screen will display the names of the variables which need initial conditions and parameter values. Numerical values are entered by typing (NAME) = (VALUE) on the last line of the screen. Numerical values of the integration method may be changed as often as desired.

When finished entering data, the user simply types RUN on the command line. The screen next flashes

PLEASE SELECT NUMBER OF
OUTPUT POINTS. MAX 512

then

PLEASE SELECT DISPLAY

and finally

DIAL COMPUTER 3243

The first two requests are used to minimize the length of time the user has to wait for output data to be transmitted to the local terminal. The first request requires a number between 10 and 512. The user responds to the

second request by typing, say:

DISPLAY (TIME) X1, X2, BETA, T

where X1 and X2 are state variables, BETA is an output variable and T is time. The user may select up to 5 variables in any output request. The file name, in this case TIME, is required in all output requests in order to distinguish between the four output storage files available to the DARE III user. All output data stored during the simulation study will always be available on the 6400 for later retrieval, regardless of the data returned to the local terminal.

After the simulation study is finished the message

HANG UP THE PHONE

will once again appear on the screen. The display selected above will flash on the XY display screen when the telephone has been disconnected. Scale factors, etc., will appear on the alphanumeric screen. A tabulation of the current graphic data precise to only three places can be obtained by typing:

QLIST X1, X2, BETA

or

QPRINT X1, X2, BETA

More precise tabulations require an additional DARE III/6400 access.

As in DARE I, additional output requests can be made at any time. The output requests may require an additional DARE III/6400 access, in which case the screen will once again flash

DIAL COMPUTER 3243

The local terminal will, however, always try to complete the output request without an additional DARE III/6400 access. The graphic data returned from the additional 6400 accesses will destroy the data currently available on the local disk. The destroyed data will, however, still be available on the 6400 storage files.

The use of the 6400 line printer for long tabular listings is encouraged, especially since the local teletypewriter is extremely slow. The line-printer listings can be picked up at the computer center under the job name DARE3. The user's ability to create long output listings without any input deck will continuously amaze the I/O clerks.

DISCUSSION—THE DARE IV SYSTEM.

The 2,000 bits/sec data rate of the inexpensive, unconditioned, dial-up telephone line is just sufficient to

return oscilloscope displays at low audio frequencies. Transmitting a reasonably large program, say, 600 character lines takes about 4 minutes, which is still tolerable. The main delay is in getting access to a CDC 6400 control point via the user queue of the system, which is, at heart, still a batch-processing system. When the 6400 was busy, delays up to 15 minutes were experienced, and such delays in an interactive simulation are somewhat hard on the nerves.

With DARE III, the user might need, moreover, three such accesses for a single computer run: once to submit the program, once to enter parameters and execute the simulation study, and once to get extra solution displays, if any. This problem can be greatly relieved by acquiring a high input-queue priority, say by crossing the computer-center administration's palms with money. Since we had no money, ours proved to be incorruptible.

A viable alternative is to change the apportionment of tasks between the large central processor and the local minicomputer. If the precompiling (translation) of the CSSL programs can be done in the local minicomputer, the latter will find the undefined parameters in the sorting process, flash them on the CRT screen, and accept the operator's parameter entries. The translated program and parameter values can then go

to the central computer in a single access. The central computer still compiles the program and proceeds to solve the differential equations. In most applications, we will then require only one access to the remote central processor, a very great advantage.

COMPUTING SPEEDS

For a typical medium-sized aerospace simulation problem involving second-order Runge-Kutta integration of 12 state-variable derivations, 100 sums, 140 products, 10 sine-cosine evaluations, and 8 table-lookup functions of one variable, the floating-point DARE I system can accommodate sinusoidal oscillations up to about 0.1 Hz, while DARE II (fixed point) goes to 4Hz, and DARE III (floating-point) will admit 7Hz.¹ The CSSL benchmark problem (pilot ejection problem)⁶ takes 64 sec for 815 runs with DARE III. Digital computing times will be proportionately longer in larger simulation problems. While on-line digital simulation does not match the bandwidth of the latest analog computers, all-digital real-time simulation is possible for many practical problems, and problem setup and checkout is incomparably simpler for digital simulation. Analog/hybrid computation will hold its own mainly in problems requiring a very large number of simulation runs on large systems, and in certain high-speed Monte Carlo, optimization, and partial-differential-equation studies.¹

DISPLAY AND CONSOLE REQUIREMENTS FOR ON-LINE SIMULATION

At the present time, Project DARE employs a television-raster alphanumeric display with internal memory, plus an 11-inch electrostatic-CRT graphic display refreshed by memory interlace from the PDP-9 almost without programmed instruction. Packed 18-bit computer words simultaneously transfer the X and Y coordinates of each displayed point to save refresh time and memory.⁷ There is also a separate simple color display.⁸

It is clearly desirable to minimize the equipment committed to each local time-sharing station. The minimal display facility would consist of a simple storage-tube display for both alphanumeric and graphic output. Such a display has excellent resolution and saves local computer time, but will not permit quick on-line editing of alphanumeric text. The DARE CRT editor program is so very convenient that it is well worth the extra cost of a television-raster alphanumeric display (CRT typewriter). Such units incorporate simple refresher memories (usually MOS shift registers)

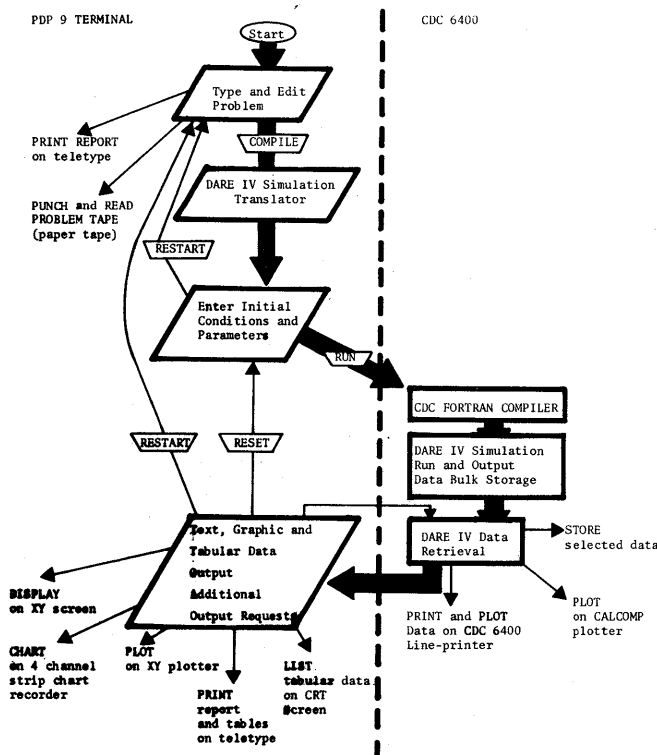


Figure 4—Time sequential flow chart of DARE IV

```

* DERIVATIVE BLOCK:
* SAMPLE PROBLEM—PILOT EJECTION
*
* VE = SEAT EXIT VEL.
* THE = SEAT EXIT ANGLE
* Y1 = HEIGHT OF RAILS
*
  X' = V*COS(TH) - VA
  Y' = V*SIN(TH)
  PROCED YGEY1 = Y, Y1
  YGEY1 = 1.
  IF(Y.LT.Y1)YGEY1 = 0.
  ENDPRO
  V' = -YGEY1*(D/AM + G*SIN(TH))
  TH' = -YGEY1*(G*COS(TH))/V
  D = RHOP*V**2/2.
  YF = Y
  TERMINATE X+30.
  AM = 7.
  G = 32.2
  DISPLAY Y, X
* LOGIC BLOCK:
* VA = PLANE VEL.
*
  H = 0.
  S = 10.
  CD = 1.
  INPUT VA, VE, THE
  OUTPUT H
* CONVERT THE TO RADIANS
  THE = THE/57.2957795
*
* CALCULATE INITIAL PILOT VEL.
*
2  V = ((VA - VE*SIN(THE))**2 + (VE*
  $ COS(THE))**2)**0.5
*
* CALCULATE INITIAL PILOT ANGLE
*
  TH = ATAN(VE*COS(THE)/(VA - VE*
  $ SIN(THE)))
*
3  RHOP = RHO(H) * CD * S
  VS = V
  THS = TH
  VAS = VA
  CALL SHOW (H, VAS, VS, THS)
  CALL RUN
  IF (YF.GT.20.) CO TO 4
  H = H+500.
  GO TO 3
*
4  RUNNO = VA
  CALL STORE
  VA = VA+50.
  IF (VA.LE.1000.) GO TO 2
* OUTPUT BLOCK:
DISPLAY H
* TABLE BLOCK NO. 1:
  RHO, 12
  0., 2.377E-3
  1E3, 2.303E-3
  2E3, 2.241E-3
  4E3, 2.117E-3
  6E3, 1.937E-3
  10E3, 1.755E-3
  15E3, 1.497E-3
  20E3, 1.267E-3
  30E3, 0.391E-3
  40E3, 0.587E-3
  50E3, 0.364E-3
  60E3, 0.2238E-3
* DATA:
DT = 1.0E-01
TMAX = 2.0E+00
X =
Y =
V =
TH =
RHOP =
VE = 40
VA = 100
Y1 = 4
THE = 15

```

Figure 5—Problem listing for pilot ejection problem

and their prices have recently come down into the \$2,000-region.

The graphic display could still use a storage tube. Since storage tubes permit comparison of current and past displays stored on the screen, one could dispense with the moving repetitive solution displays possible with DARE II. For a larger display presentation, we are also considering a storage-tube/scan-converter system, which would combine the refresher-memory output of a television-scan alphanumeric-display generator with scan-converter pickup from a small storage tube on one large television screen, possibly in color.

The simulation console will also need a local mini-computer for editing, communication control, and display operation. With the storage-tube graphic display, 9- or 10-bit display-point X and Y coordinates could be stored separately, so that the local mini-computer need only be one of the new inexpensive 12-bit types costing under \$5,000 or for central processor: we would anticipate a need for at least 8K of local memory instead of the minimal 4K.

For hard-copy preparation, we have employed a teletypewriter, a handheld Polaroid oscilloscope camera capable of photographing both the alphanumeric and graphic displays, a four-channel strip-chart recorder, and an XY servo recorder. A small line printer with full 140 character lines would be faster and more reliable than our KSR 35 teletypewriter. Unlike the latter, the line printer could accept the full-width of the 6400 output for debugging; the line printer could also be used for plotting solutions. Very extensive line-printer tables and CALCOMP graphical plots could also be prepared at the computer center.

DIGITAL-COMPUTER ARCHITECTURE FOR SIMULATION

Most modern 24- to 36-bit intermediate-sized digital computers with floating-point arithmetic are well suited for simulation applications. Since no such machine is available at the University of Arizona, the DARE IF project will investigate the augmentation of an existing 18-bit minicomputer (PDP-9 or PDP-15) with a newly designed floating-point arithmetic unit plus some high-speed storage. The result will be a new small general-purpose computer, but we are, of course, especially interested in those features of digital-computer architecture which might favor continuous-system simulation.

The very fast MECL-II emitter-coupled logic (2 to 3 n sec gate delay) was chosen for the new processor to permit us to trade this speed for a relatively simple arithmetic design. The fast processor can communicate with the PDP-9 memory through the direct-memory access channel, which requires 1 μ sec for the transmission of an 18-bit instruction, or 3 μ sec for the transmission of a 54-bit floating-point data word (consisting of three 18-bit PDP-9 words). These word-transfer rates are fairly well matched to the anticipated 10 μ sec to 15 μ sec floating-point addition and multiplication times in the fast arithmetic unit. We will, nevertheless, investigate instruction lookahead and the use of some fast-storage (scratchpad memory) consisting of MECL-II memory chips to buffer some data and/or instruction transfers in an effort to match the arithmetic processor's speed to better advantage.

A look at a typical simulation program indicated that most of the program execution involves the repetitive calling of derivative-computing, integration-formula,

and data-storing subroutines. It would appear that many time-consuming core accesses could be saved through storage of complete subroutines in a fast scratchpad memory. Additional execution time would be saved if instruction, fetching, and/or data storage could be overlapped with arithmetic execution.

Further investigation of derivative computations for differential-equation solution indicates that the requirements for intermediate storage are relatively small. The implementation of a typical simulation block diagram requires one word of temporary storage for each point where the block-diagram interconnections branch. Fast-access storage in multiple arithmetic registers, small scratchpad memories, or special memory stacks would appear to be especially suited to such operations and could save many time-consuming core accesses. Indeed, the organization of derivative computations tempts the designer to organize his scratchpad storage into a stack for temporary data storage. A stack-oriented processor would permit a wide variety of 18-bit operating instruction, with only a minimum of memory-reference instructions for communicating with core storage. Unfortunately, such a stack organization of the fast processor would also increase the total number of instructions (and instruction fetches!) required for the total derivative-computing program. The optimal system would have enough fast scratchpad storage to store all frequently used subroutines, but this is a fairly expensive proposition; a future DARE study (DARE IIF, Table 1) will look into possible compromises and trades.

SOME CONCLUSIONS

We believe that the success of the DARE I and DARE III (equation-oriented) and DARE II (block-oriented) simulation systems has conclusively proved the feasibility and advantages of all-digital on-line simulation. Without question, all future systems of this type will be scale-factor free floating-point systems. In our experience, most users appear to prefer the equation-oriented systems. On the other hand, the possibility of creating special frequently used system blocks as macros is a very convenient feature of the block-oriented DARE II language. Future continuous-system-simulation systems will superimpose a macro generator on equation-oriented systems; this has already been done in the batch-processed CSMP-360 and in some of the newer CSSL systems. In the final analysis, the main advantage of assembler-based purely block-oriented simulation systems will depend on the extent of their execution-speed advantage over compiler-based equation-oriented systems. This speed advantage is

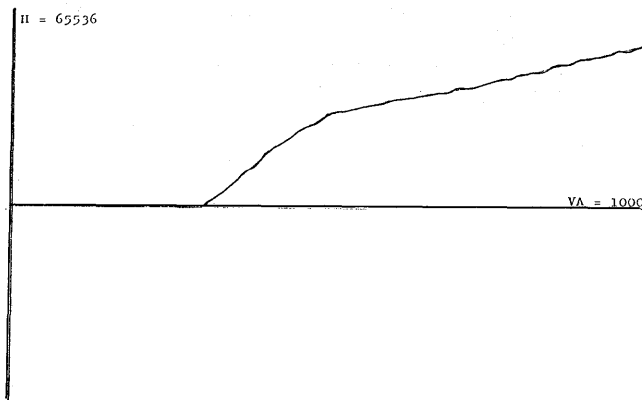


Figure 6—Plot of H vs. VA for pilot ejection problem

TABLE I—Project DARE On-line Simulation Systems

System	Author	Completed	Description	Computer	Reference
DARE I	J. Goltz	1969	Floating-point, equation-oriented CSSL System, 20 state variables one derivative block	PDP-9	1, 2
DARE IR	J. Moore	1971	Similar to DARE I, two derivative blocks	PDP-9	
DARE II	T. Liebert	1970	Fixed-point, block-oriented system two derivative blocks, extremely fast	PDP-9	1, 3
DARE III	H. Aus	1971	Floating-point, equation-oriented CSSL time-sharing system, 200 state variables, two derivative blocks	PDP-9 CDC 6400	4
DARE IIIB	A. Trevor	1971	Batch-processed CSSL, 200 state variables, two derivative blocks	CDC 6400	5
DARE IF DARE IIF	C. Wiatrowski	1972	Similar to DARE IR Similar to DARE II, but floating-point	PDP-9 and homemade floating- point processor	
DARE IV			Modified version of DARE III	PDP-9 CDC-6400	

overwhelming with minicomputers which, because of their small memory sizes, rely on many subroutine calls for FORTRAN execution. With large digital computers having larger core memories and floating-point hardware, together with modern, very efficient FORTRAN compilers, much of the speed advantage of assembler-based systems may be lost. Estimates of this remaining speed advantage vary, but might be in the order of two-to-one, which would still result in significant cost savings.

ACKNOWLEDGMENTS

Project DARE is sponsored by the National Science Foundation under NSF Grants GK-1860 and GK-15224. DARE I and DARE II were respectively written by John Golts (now President of COMPU-SERVE, Columbus, Ohio) and Tom Liebert (now on the technical staff of the Bell Telephone Laboratories) as Ph.D. dissertations. Professor John V. Wait is co-principal investigator.

REFERENCES

- 1 G A KORN
Project DARE: Differential analyzer replacement by on-line digital simulation
Proceedings Fall Joint Computer Conference 1969
- 2 J R GOLTZ
The DARE I simulation system
Proceedings SWIEEEO Dallas Texas 1970
- 3 T A LIEBERT
The DARE II simulation system
Proceedings SCSC Denver Colorado 1970
- 4 H AUS
DARE III, A time-shared digital simulation system
PhD Dissertation University of Arizona 1971
- 5 A TREVOR
The DARE IIIB simulation system
M S Thesis University of Arizona 1971
- 6 *The SCI continuous-system simulation language*
SCI Software Committee Simulation December 1967
- 7 G A KORN et al
A new graphic display/plotter for small digital computers
Proceedings Spring Joint Computer Conference 1969
- 8 C WIATROWSKI
A color television graph plotter
Computer Design April 1970

A panel session—Computer structure—Past, present and future

Possibilities for Computer Structures 1971*

by C. GORDON BELL and ALLEN NEWELL

Carnegie-Mellon University

What computer structures come into existence in a given epoch depends on the confluence of several factors:

The underlying technology—its speed, cost, reliability, etc.

The structures that have actually been conceived.
The demand for computer systems (in terms of both economics and user influence).

One ignores any of these factors at one's peril. In particular, with technology moving rapidly, a real limitation exists on our ability as designers to discover appropriate structures that exploit the new trade-offs between the various aspects of a computer system.

The design of computer structures is not a systematic art. So new is it, in fact, that in a recent book (Bell and Newell, 1971) we found ourselves dealing with basic issues of notation. We are still a long way from concern with the sort of synthesis procedures that characterize, say, linear circuit design. However, the immaturity is dictated, not so much by youth (after all we have been designing computers for almost 30 years), as by the shifts in technology that continually

* The ideas expressed in this presentation have emerged from a number of overlapping design efforts, mostly around CMU and DEC, but occasionally elsewhere (e.g., at Newcastle-on-Tyne, the ARPA list processing machine effort, and the effort at the Stanford AI project). Consistent with this being a short note, we have attempted to indicate the individuals involved in these efforts at appropriate places in the text. But we wish here to acknowledge more generally the contribution of all these individuals. The preparation of this paper was supported by the Advanced Research Projects Agency of the Office of the Secretary of Defense (F44620-70-C0107) and is monitored by the Air Force Office of Scientific Research. The paper is to be published in the *Proceedings of the FJCC, 1971* and may not be copied without permission.

throw us into previously uninhabited parts of the space of all computer structures. Whatever systematic techniques start to emerge are left behind.

This note comments on several possibilities for computer structures in the next half-decade. Given the unfamiliarity that we all have with the region of computer space into which we are now moving, there can be no systematic coverage. Neither is it appropriate simply to reiterate what would be nice to have. Such an exercise is not responsive to the new constraints that will limit the new designs. Such constraints will certainly continue to exist, no matter how rapidly logic speed rises and logic costs fall. In fact, it is useful to view any prognostication of new computer structures (such as this paper) as an attempt to reveal the nature of the design constraints that will characterize a new epoch of technology.

We will discuss five aspects of computer structures. Mostly, these represent design features that we think have a good possibility of becoming important in the next few years, though we have reservations on one. We have been actively engaged (with others) in working on particular structures of the type we present. Our selection of these is not a denial that other quite different structures might also be strong contenders for dominance during the next several years. Indeed, according to the point made earlier, with strong shifts in technology no one can know much about the real potentialities for new structures. Thus, that we have been working on these particular structures provides, mainly, a guarantee that we have thought hard enough about their particulars to have some feeling for the design limitations in their local vicinity.

Minicomputer multiprocessor structures

Consider the multiprocessor structure of Figure 1. There are p central processors (P_c) and m primary memories (M_p). We ignore, in this discussion, the remaining structure that connects the secondary memories and i/o. The switch (S_{mp}) is effectively a crossbar, which permits any of the processors access to any of the memories.

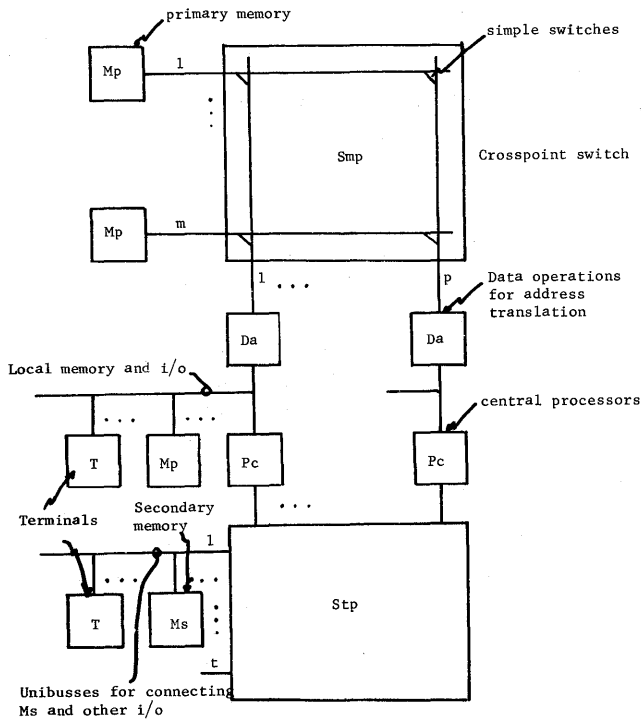


Figure 1—Smp (crosspoint) for connecting p central processors (Pc) from primary memories (Mp)

There is nothing new per se about a multiprocessor structure. Many dual processors exist, as do genuine multiprocessors whose additional processors (beyond one Pc) are functionally specialized to i/o and display. General multiprocessors have been proposed and a very few have come into existence (e.g., the Burroughs D825). But they have not attained any substantial status. The main technological reasons appear to be (1) the cost and reliability of the Smp and (2) the relative cost of many processors. Software (i.e., operating systems) is also a critical difficulty, no doubt, but not one that appears yet to prohibit systems from coming into existence.

Both of these technical factors appear to be changing sufficiently to finally usher in multiprocessor systems of substantial scope. The cost of the processor is changing most rapidly at the minicomputer end of the scale. Thus, we expect to see minicomputer multiprocessors systems before those with large work-length Pc's. An additional impediment for large Pc's is the bandwidth required through the switch, which is substantially less for 16b/w machines than for 32-64b/w machines both in terms of cost and reliability.

As a basis for discussing detailed technical issues, let us describe a multiprocessor system involving the DEC PDP-11. Variant designs of this system have

been proposed both at CMU and at Newcastle-on-Tyne.* A set of p PDP-11's have access to a set of m Mp's aggregating 2^{21} 8b bytes.** Each Pc maintains its address space of 2^{16} bytes, but an address mapping component (Da) associated with each Pc permits this address space to be distributed as 2^8 independent pages of 2^{13} bytes each. The details of this addressing, though important, need not be discussed here. Similarly, the details of the Smp need not be discussed. Each link through the Smp is essentially a unibus (the bus of the PDP-11, see Bell et al., 1969). Connections are made on a memory access basis, so that the a PC broadcasts d's address to all Mp's and the connection is made to the recognizing Mp for the data transfer.

The three critical questions about the Smp are its performance, measured in terms of Pc effectiveness, its reliability and its cost. Figure 2 gives the calculated expected performance (Strecker, 1970) in terms of total effective memory cycle access rate of the Pc's (whose number is shown along the abscissa). Each instruction

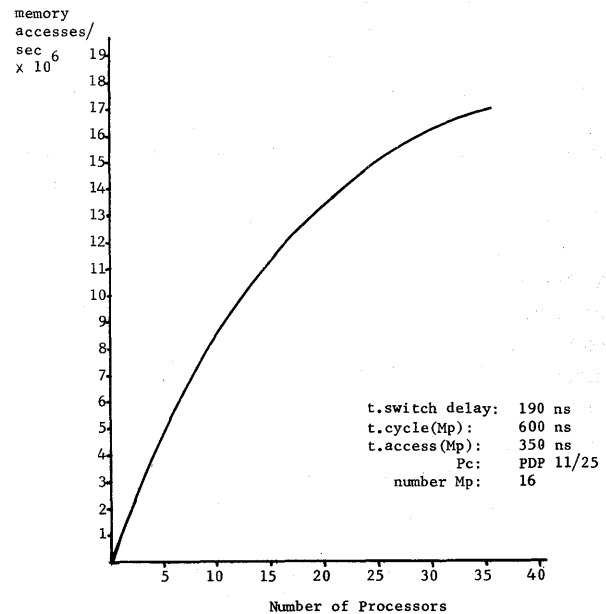


Figure 2—Performance of a multiprocessor computer with 16 independent Mp's.

* The original design was proposed by W. Wulf and W. Broadley, based on a switch design by Bell and Broadley; a second more general design was proposed by C. G. Bell, H. Lauer and B. Randall at Newcastle-on-Tyne; the version described here is by C. G. Bell, W. Broadley, S. Rege and W. Wulf. No published descriptions are yet available on any of the designs, though some are in preparation.

** Addressing in the PDP-11 is by bytes, though it is preferable to view it as a 16b machine.

requires one to five memory accesses. The curve is parameterized by the number of Mp's ($m=16$ here), the t_{cycle} of the Mp (350 ns here) and the delay through the switch (190 ns here). The criteria we have used for ideal performance is p stand-alone computers with no switching delays. Thus, the loss is due to both switching delay and multi-Pc interference. The parameters shown are attainable with today's technology. The number of memory references per processor decreases as the number of processors increase, since the calculation assumes a reference to any Mp is equally likely. The reliability cannot yet be estimated accurately, but appears to be adequate, based on a component count. The cost per Pc is of the order of one quarter to one times the Pc, measured in amounts of logic for a 16×16 switch. Thus, the Smp cost is appreciable, but not prohibitive.

What does one obtain with such a structure? Basically, Pc cycles have been traded for (1) access to a larger memory space and (2) Mp-level interprocessor communication. These benefits come in two styles. Statistically, the Smp permits configuration of the Pc's with various amounts of memory and isolation. An important design feature, not stressed above, is that the PDP-11 components remain essentially unmodified, so that they can be moved in and out of the system at will. This feature extends to permitting the addition and extraction of components to the system while in operation. Dynamically, the Smp permits the set of processors to cooperate on various tasks and to decrease the system overhead for input/output and operating systems programs. Coupled with this is common access to the secondary memory and peripheral parts of the systems, permitting substantially lower total system cost as opposed to p independent systems.*

Caches for multiprocessors

A key design parameter in multiprocessor organizations, such as the one above, is the delay through the switch, measured relative to the performance of the Mp's and Pc's. The total instruction (e.g., for a memory access instruction) of a Pc can be partitioned as:

$$t_{\text{instruction}} = t_{\text{Pc}} + t_{\text{Smp}} + t_{\text{Mp}}$$

In current memory technology overlap is possible between Pc and Mp since accessed information is available before the rewrite cycle is completed. How

* If this latter goal were all that were required, then one might consider less expensive alternatives. However, a price must be paid in system overhead for less general coupling and the trade-off is far from clear. In fact, we are not justifying the design here, but simply presenting a concrete example.

much this can be exploited in a multiprocessor depends on t_{Smp} . Thus, the relevant t_{Mp} is that which would obtain in a non-switched system.

Current technology makes all the above terms comparable, from 50~500 nanoseconds. Thus, variations of a factor of 2 in any of the component terms can have a determining effect on the design. Most important here is that t_{Smp} can easily become large enough to make $t_{\text{instruction}}(\text{with Smp})$ twice $t_{\text{instruction}}(\text{without Smp})$.

The cache appears to offer a solution to this problem within the currently emerging economic design parameters. The basic concept of a cache is well established.* To review: a cache operates by providing a small high access content addressed memory (M.cache) for recently accessed words. Any reference to Mp first interrogates M.cache to see if the information is there, and only if not is an access made to Mp. The basic statistical regularity of system performance underlying the cache is that words recently accessed will be accessed again. This probability of reaccess depends of course on the size of the past maintained. Available statistics show that if a few thousand words of cache can be kept, then well over 90 percent of the Mp accesses will be found in the cache, rather than having to go to Mp itself. If technology provides a steep trade-off between memory size, memory cycle time and cost per word, then a cache is a valuable structure.

If we associate the cache with the Pc, as in Figure 3, then the net effect of the cache is to decrease t_{Pc} (for fixed computational power delivered). In organizations

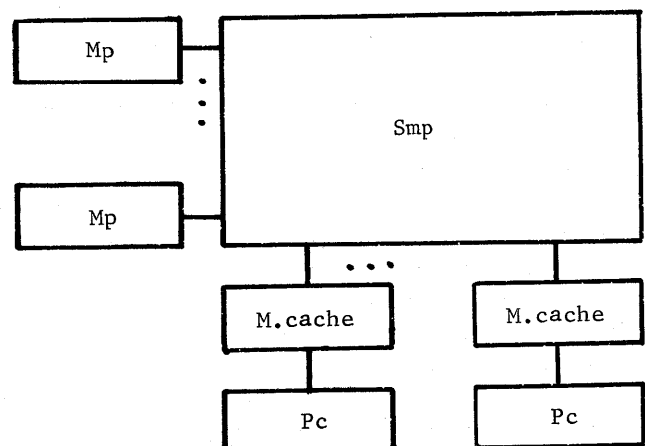


Figure 3—Multiprocessor computer with cache associated with each Pc.

* The first machine really to use a cache was the 360/85 under the name of "buffer memory" (Conti, 1969). Wilkes (1965) termed it the "look-aside" memory. "Cache" seems by now an accepted designation.

such as the 360/85 this permits balance to be achieved between a fast Pc and a slower Mp. In the case of multiprocessor, this permits the delay of Smp to be of less consequence (for aggregated t.Smp and t.Mp play the same role as does t.Mp in a uniprocessor system).

There is a second strong positive effect of caches in a multiprocessor organization of the kind under discussion. As the graph of Figure 2 shows, performance is a function not only of the delay times, but of the frequency of accessing conflicts. These conflicts are a monotone function of the traffic on the switch, increasing sharply as the traffic increases. The cache on the Pc side of switch operates to decrease this traffic, as well as to avoid the delay times. There is one serious problem regarding the validity of the data in a system such as Figure 3, where multiple instances of data co-exist. In a system with p caches and an Mp, it is conceivable that a single address could be assigned $p+1$ different contents. To avoid this problem by assuring a single valid copy would appear to require a large amount of hardware and time. Alternatively, the burden might be placed on the operating system to provide special instructions both to dump the cache back into Mp and to avoid the cache altogether for certain references.

In a recent attempt to design a large computer for use in artificial intelligence (C.ai), we considered a large multiprocessor system (Bell and Freeman, 1971; Barbacci, Goldberg and Knudsen, 1971; McCracken and Robertson, 1971).^{*} The system is similar to the one in Figure 1; in fact, the essential design of the Smp for the minicomputer-multiprocessor came from the C.ai effort. C.ai differs primarily in having 10-20 large Pc's with performance in the 5×10^7 operation/sec class (e.g., the cache-based Pc being designed at Stanford, which is aimed at $10 \times Pc$ (PDP-10) power). An essential requirement for this large multiprocessor was the use of caches for each Pc in the manner indicated.

Why then are caches not needed on the minicomputer-multiprocessor? Interestingly enough, there are three answers. The first is that the performance of minicomputers is sufficiently low, relative to the switch and the Mp, so that reasonable throughput can be obtained without the cache. The second is that the first answer is not quite true for the PDP-11 Pc. To achieve a reasonable balance in our current design requires an upgrading of the bus driving circuits on the

PDP-11.^{**} The third answer is that the benefits that accrue from a cache in fact hold for minicomputers as well. Recently a study by Bell, Cassasent and Hamel (1971) showed that a system composed of a cache and a fast minicomputer Pc was able to attain a fivefold increase in power over a PDP-8. The cost of the cache was comparable to the Pc, yielding a substantial net gain (i.e., for a minimal system the power increased by 5 while the cost doubled). Thus, caches would undoubtedly further improve the design of Figure 1 at a lower cost. Alternatively, one could simply add more Pc's, rather than increase the cost of the Pc by a cache.

Multiple cache processors

One additional design feature of the C.ai is worth mentioning, in addition to its basic multiprocessor structure and cache structure vis-a-vis the Smp.

The general philosophy of the multiprocessor is that of functionally specialized Pc's working into a very large Mp. In the context of artificial intelligence, functional specialization of the entire Pc to a completely specific system (such as the language, Lisp) seems required to exploit algorithm specialization.^{*} Thus, we engaged in the design of two moderate sized Pc's, one for Lisp and one for a system building system called L* (Newell, McCracken, Robertson and DeBenedetti, 1971).

Figure 4 shows the basic PMS organization of one of these processors (actually the one of L*, but it makes little difference to the discussion at hand). The important feature is the use of multiple caches, one for data and one for the microprogram. Two gains are to be obtained from this organization. On the performance side, the gain is essentially a factor of 2, arising from the inherent parallelism that comes from the lockstep between the data and instruction streams. The cache is indicated by the design decision to permit the microcode to be dynamic. Thus, the second gain is in replacing a deliberate system programming organization for changing the microcode with the statistical structure of the cache, thus simplifying considerably the total system organization (including the operating system).

^{**} Modification of these circuits constitutes the primary modification of the PDP-11 Pc for participation in the system. The only other modification is the use of two bits in the program status work to indicate extended addressing.

^{*} The argument is somewhat complex, involving the fact that specialization to artificial intelligence per se (and in particular to list processing) does not produce much real specialization of hardware. Not until one moves to a completely particular specification of internal data types and interpretation algorithms can effective specialization occur.

^{*} Many people at CMU participated in the C.ai effort; a list can be found in the reports referenced. Furthermore, the C.ai effort was itself imbedded in a more general design effort initiated by the Information Processing Technology Office of ARPA and was affected by a much wider group.

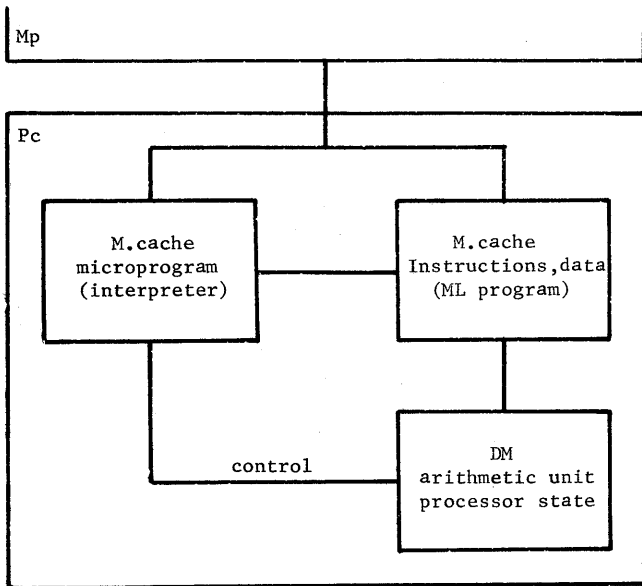


Figure 4—Multiple (two) cache system

The gains here are not overwhelming. But in the light of the many single cache organizations (Conti, 1969) and non-cache dynamic microprogramming organizations (Husson, 1970; Tucker and Flynn, 1971) being proposed, it seems worth pointing out. The concept could be extended to more than two caches in computers that are pipelined, where additional parallelism is available in the controls.

Register transfer modules

Some time ago Wes Clark (1967) proposed a system of organization that he called Macromodules. These traded speed and cost to obtain true Erector set constructability. For a given domain of application, namely sophisticated instrument-oriented laboratory experimentation, a good case could be made that the trade-off was worthwhile. The modules essentially incorporate functions at the register-transfer level of computer structure, thus providing a set of primitives substantially higher than the gates and delays of the logic circuit level.

More recently, another module system has been created, called Register-Transfer-Modules (RTM'S)* (Bell and Grason, 1971). RTM's differ from Macromodules at several design points, being cheaper (a factor of 5), slower (a factor of 2), harder to wire, and more permanent when constructed. On some dimensions (e.g., checkout time) not enough evidence is yet available. Thus, they occupy a different point in a design

space of RT modules. For our purposes here these two systems can be taken together to define an approach to a class of computer systems design.

Register transfer modules appear to be highly effective for the realization of complex controls, e.g., instrument controls, tape and disk controls, printer controls, etc. They appear to offer the first real opportunity for a rationalization of the design of these aspects of computer systems. Their strong points are in the rationalization of the control itself and in the flexibility of data structures.

An extremely interesting competition is in the offing between minicomputers and register transfer modules.* As the price of the minicomputer continues to drop, it becomes increasingly possible simply to use an entire C.mini for any control job. The advantages are low cost through standardization and hence mass production. To combat this the modular system has its adaptation to a particular job, especially in the data flow part of the design, thus saving on the total amount of system required and on the time cost of the algorithm.

An important role in this competition is played by memory. If substantial memory is required, its cost becomes an important part of the cost of the total system. An Mp essentially requires a Pc and lo! a minicomputer has been created. Stated another way: a minicomputer is simply a very good way to package a memory. Consequently, RT modules cannot compete with minicomputers in a region of large Mp. This extends to task domains that require very large amounts of control, since currently a memory is the most cost effective way to hold a large amount of control information. Thus, the domain of the RT modules appears to be strongly bounded from above.

An interesting application of the above proposition can be witnessed in the domain of display consoles. First, substantial memory is required to hold the information to be displayed. Thus, in essence, small computers (P.display-Mp) have been associated with displays. A few years back costs were such as to force time-sharing; each P.display serviced several scopes. But the ratio is finally coming down to 1-1, leading to simplification in system organization, due to the elimination of a level of hierarchical structure.

Our argument above, however, has a stronger point. Namely, a minicomputer (namely, a Pc-Mp organization) will dominate as long as there is already the requirement for the memory. Thus, the specialized display processors are giving way to general organizations. In fact, it is as effective to use an off-the-shelf mini-

* Also called the PDP-16 by DEC.

* Actually there may be a third contender, microprogrammed controllers.

computer for the display processor as one specially designed for the purpose. Our own attempt to show this involves a PDP-11 (Bell, Reddy, Pierson and Rosen, 1971).

There is an additional reason for discussing RT modules, beyond their potentiality for becoming a significant computer structure. They appear to offer the impetus for recasting the logic level of computer structure. The register transfer level has slowly been gathering reality as a distinct system level. There appears to be no significant mathematical techniques associated with it. But in fact the same is true of the logic level. All of the synthesis and analysis techniques for sequential and combinatorial circuits are essentially beside the point as far as real design is concerned. Only the ability to evaluate—to compute the output given the input, to compute loadings, etc.—has been important. Besides this, what holds the logic level intact is (1) a comprehensible symbolism, (2) a clear relation of structure to function so a designer can create useful structures with ease, and (3) a direct correspondence between symbolic elements and physical elements.

RT modules appear to have the potential to provide all three of these facilities at the register transfer level (rather than the sequential and combinatorial logic level). The ability to evaluate is already present and has been provided in several simulators (e.g., Darringer, 1969; Chu, 1970). The module systems provide the direct correspondence to physical components, which is the essential new ingredient. But there is also emerging a symbolism with clear function-structure connections, so that design can proceed directly in terms of these components. For the Macomodules of Clark one can actually design directly in terms of the modules. With our RTMs we have been able to adapt the PMS notation (Bell and Newell, 1971) into a highly satisfactory symbolism* It is too early to see clearly whether this conceptual event will take place. If it does, we should see the combinatorial and sequential logic levels shrink to a smaller, perhaps even miniscule, role in computer engineering and science. Actually, even if these modules do not cause such an emphatic shift in digital design, it is almost safe to predict this change solely on the basis of minicomputers and microprogrammed controllers being used for this purpose. This will lead to a decrease in the need for, and interest in, conventional sequential and combinatorial logic design.

A cautionary note on microprogramming

With the right shaped trade-off function on memory speeds, sizes and costs relative to logic, microprogram-

ming becomes a preferred organization, because of the regularity in design, testability and design flexibility that it offers. Memories of 10^5 bits must be available at speeds comparable to logic and at substantially lower cost per effective gate. With only 10^4 bits there is not enough space for the microcode of a large Pc. If memory is too slow or too costly, the resulting Pc's simply cannot compete with conventional hardwired Pc's in terms of computational-power/dollar.

The conditions for microprogramming* first became satisfied with read-only memories (circa 1965). In the first major experiment, the IBM System/360, a variety of hardware was used at different performance levels of the series, all of it M.ro. Some of the memories permitted augmentation, and in fact this feature attained some significant use, e.g., the RUSH system of Alan Babcock (a Joss-like commercial timesharing system based on PL/I) which is able to be both cost-effective and interpretive by putting parts of the interpreter into the M.microprogram of the 360/50.

More recently read-write memories have become available at speeds and costs that satisfy the conditions for microprogramming. This leads, almost automatically, to dynamic microprogramming, in which the user is able to modify the microcode under program control. This allows his program to be executed at higher speeds. The effect is not quite to make the microcode the new machine language, for the trade-offs still do not permit $10^6 \sim 10^7$ bits of M. μ p, which is required for full sized programs. Thus, the original functional concept of microprogramming remains operative: a programmed interpreter and instruction set for another machine language, which occupies a much larger Mp.

All this story is a rather straightforward illustration of the principle that computer structures are a strong function of the cost-performance trade-offs within a given set of technologies. Different regions in the space of trade-offs lead, not to parametric adjustments in a given invariant computer structure, but to qualitatively different structures.

The cautionary note is the following. In our headlong plunge to discover the new organizations that seem to be effective in a newly emerging trade-off region, we must still attempt to separate out the gains to be made from the various aspects of the new system—from the new components, from the newly proposed organizations, etc. The flurry of work in dynamic microprogramming seems to use to be suffering somewhat in this regard. The proposed designs (e.g., see Tucker and Flynn, 1971) appear to be conventional minimum

* Called Chartran in DEC marketing terminology. See Bell and Grason (1971) for examples.

* The microprocessor must operate at a speed of 4 to 10 times the processor being interpreted.

computers with wide unencoded words.* They compare very favorably against existing systems (e.g., members of the 360 series), but when the performance gains are dissected they appear to be due almost entirely to the gains in componentry, rather than to any organizational gains (e.g., Tucker and Flynn, 1971).* The cost of these systems is usually missing in such analyses.

	High performance technology microprocessor	Model 50	Mini- computer
number of instructions	8	11 (6)	5
number of bits	512	224 (128)	80
loop time (in memory accesses)	10	9 (6)	7
memory bandwidth (megabits/sec)	640~3840	16	16
time for 10 iterations (μ sec)	4.3	191	70
time using high performance technology (μ sec)	4.3	6.5 (4)	3.5

() indicates improvement in coding over Tucker and Flynn.

*There appears also to be some confusion in the application of the term "microprogram" to some of the proposed systems. The definition given by Wilkes (1969) is functional: a microprogrammed Pc is one whose internal control is attained by another processor, P.microprogram. Thus, it is the cascade of two processors, one being the interpreter for the other. Certain structural features characterize current P.microprograms: wide words; the nature of the operations (control of RT paths); parallel evocation of operations, and explicit next-instruction addressing (to avoid machinery in the P.microprogram). Many of the proposed dynamic programming systems maintain some of the structural features, e.g., wide words, but drop the functional aspect. This is, of course, essentially a terminological matter. However, we do think it would be a pity for the term microprogramming to attach to certain structural features, independent of function, rather than to the functional scheme of cascaded processors, one the interpreter for the other.

*A microprogrammed processor design using 1971 logic and memory technology was compared with IBM's 1964 Solid Logic Technology and core memory used in the 360 Model 50. Since the newer technology (50 nanoseconds/64 bits) was a factor of about 80 faster than the Model 50 (2000 nanoseconds/32 bits) the microprogrammed processor was somewhat (only a factor of 45) faster. Even using the faster technology the microprogrammed processor's times for multiplication given by Tucker and Flynn were about the same as Model 50.

The following table of a Fibonacci number benchmark given by Tucker and Flynn shows that the main advantage of microprogramming is with high performance technology. A microprogrammed processor has about the same number of instructions and number of memory accesses. Due to the poor encoding of instructions a microprogram takes more bits (hence possibly costs more). By having a comparatively high memory bandwidth it can execute the loop rapidly, but given a model 50 or a mini-computer constructed with a 50 ns memory the execution times are about the same.

Actually, there are signs of the watchmaker's delusion (Simon, 1969). A watchmaker, Tempus, attempted to construct watches out of very small components, but every time the phone rang with an order he was forced to start over. He got very few watches completed. His friend, Hora, decided first to build springs, releases, escapements, gears, etc., and then larger assemblies of these. Though he, too, was often called to the phone, he quite often had time to complete one of these small assemblies, and then to put these together to obtain an entire watch.

To apply the moral: Large systems can only be built out of components modestly smaller than the final system itself, not directly out of much smaller components. The dynamic microprogramming proposals take as given the same micro-components as have existed priorly (gates and registers). They do not propose any of the intermediate levels of organization that are required to produce a large system. Thus, e.g., when they propose to put operating systems directly in microcode they are close to the watchmaker's delusion. Insofar as the response is "But of course we expect these intermediate levels of organization to exist," then their proposals are radically incomplete, since the operative concepts of the design are missing.

The situation is even a little worse, for unlike conventional machine language organizations, microprogrammed processors are usually oriented to highly special technology, have multiple automatic units that have to be operated in parallel, can even perform in a non-deterministic manner, are location sensitive, and provide a combinatorially larger instruction set. Effective compilers and performance-monitoring software will be mandatory before users can effectively gain any order-of-magnitude increase in performance latent in the basic organization. Furthermore, since these processors are so technology oriented, it is difficult to guarantee that they will have successors or be members of compatible families.

CONCLUSION

We have touched on a number of aspects of current research in computer structures that appear to have possibilities for being important structures in the next half decade. Our examples—and our style of discussing them—suggest several basic points about the design of computer structures. Some of these have been stated already in earlier sections, but it seems useful to list them all together:

- (1) Computer design is still driven by the changes in technology, especially in the varying trade-offs.

- (2) Distinct regions in the space of trade-offs lead to qualitatively different designs.
- (3) These designs have to be discovered by us (the computer designers), and this can happen only after the trade-off characteristics of a new region become reasonably well understood.
- (4) Thus, our designs always lag the technology seriously. Those that are reaching for the new technology are extremely crude. Those that are iterations on existing designs, hence more polished, fail to be responsive to the newly emerging trade-offs.
- (5) Since the development cycle on new total systems is still of the order of years, the only structures that can be predicted with even minimal confidence are those already available in nascent form. The multiprocessor, cache and RT module organizations discussed earlier are all examples of this.
- (6) The design tools that we have for discussing (and discovering) appropriate designs are weak, especially in the domain over which the structures under consideration here have ranged—essentially the PMS level.
- (7) In particular, there is no really useful language for expressing the trade-offs in a rough and qualitative way, yet precisely enough so that the design consequences can be analyzed.
- (8) In particular (as well), design depends ultimately on having conceptual components of the right size relative to the system to be constructed: small enough to permit variety, large enough to permit discovery. The transient character of the underlying space (the available space of computer structures) reinforces the latter requirement. The notion of M.cache is an example of a new design component with associated functions, not available until a few years ago. Even this small note shows it to be a useful component in terms of which designs can be sought. The potential conceptual revolution hiding in the RT modules provides another example.

REFERENCES

- M BARBACCI H GOLDBERG M KNUDSEN
A LISP processor for C.ai
Department of Computer Science Carnegie Mellon University 1971
- C G BELL R CADY H MCFARLAND B DELAGI
J O'LAUGHLIN R NOONAN W WULF
A new architecture for mini-computers—The DEC PDP-11
AFIPS Conference Proceedings Vol 36 Spring Joint Computer Conference 1970
- C G BELL D CASSASANT R HAMEL
The use of the cache memory in the PDP-8/F minicomputer
AFIPS Proceedings of the Spring Joint Computer Conference 1971
- C G BELL P FREEMAN et al
A computing environment for AI research
Department of Computer Science Carnegie-Mellon University 1971
- C G BELL J GRACON
The register transfer module design concept
Computer Design pp 87–94 May 1971
- C G BELL A NEWELL
Computer structures
McGraw-Hill 1971
- C G BELL D R REDDY C PIERSON B ROSEN
A high performance programmed remote display terminal
Computer Science Department Carnegie-Mellon University 1971
(For IEEE Computer Conference 1971)
- Y CHU
Introduction to computer organization
Prentice-Hall 1970
- W A CLARK
Macromodular computer systems
AFIPS Proceedings Spring Joint Computer Conference pp 335–336 1967 (This paper introduced a set of six papers by Clark and his colleagues pp 337–401)
- C J CONTI
Concepts for buffer storage
IEEE Computer Group News March 1969
- J A DARRINGER
The description, simulation and automatic implementation of digital computer processors
PhD dissertation Carnegie-Mellon University 1969
- S S HUSSON
Microprogramming: Principles and practice
Prentice-Hall 1970
- D MCCRACKEN G ROBERTSON
An L processor for C.ai*
Department of Computer Science Carnegie-Mellon University 1971
- A NEWELL D MCCRACKEN G ROBERTSON
L DEBENDETTI
L(F) manual*
Department of Computer Science Carnegie-Mellon University 1971
- H A SIMON
The sciences of the artificial
MIT PRESS 1969
- W Strecker
Analysis of instruction execution rates in multiprocessor computer system
PhD dissertation Carnegie-Mellon University 1970
- A TUCKER M J FLYNN
Dynamic microprogramming: Processor organization and programming
Communications of the ACM 14 pp 240–250 April 1971
- M V WILKES
Slave memories and dynamic storage allocation
IEEE Transactions on Computers Vol EC-14 No 2 pp 270–271 1965
- M V WILKES
The growth of interest in microprogramming: A literature survey
Computing Reviews Vol 1 No 3 pp 139–145

Computer Structures: Past, Present and Future (abstract)

by FREDERICK P. BROOKS, JR.

University of North Carolina
Chapel Hill, North Carolina

First, Blaauw's law of the persistence of established technology leads me to predict that both c.p.u. architecture and technology will change little by 1975, and will be surprisingly similar in 1980.

Second, magnetic bubbles or integrated circuits may at least give us memories in the 100 sec. range. These will force the complete abandonment of the fast-fading dichotomy between electronic memories, directly addressed, and mechanical memories, treated as input-output. As the memory hierarchy becomes a continuum, radically improved addressing techniques and block-moving algorithms will be required.

Third, cheap minicomputers lead one to distributed intelligence systems, with minicomputers replacing disk control units, display processors, or communications adapters. Such systems promise to save c.p.u. core, save c.p.u. cycles, and simplify programming. But first attempts have saved few c.p.u. cycles and no core, and programming is worse. The answer seems to be to combine distributed intelligence—separate instruction fetching and interpreting mechanism—with centralized memory.

Computer Structures: Past, Present and Future (abstract)

by D. B. G. EDWARDS

University of Manchester
Manchester, England

The machine being constructed at Manchester, M.U.5, has a number of interesting structural features which result from the general aim of designing a system to function efficiently with high level languages.

The first feature is a 'Paging' system which is based on Atlas experience and improved to provide a large virtual address range (34 bits), good protection facilities and an ability to simultaneously handle a number of

different page sizes. The next concept termed 'Naming' was introduced after extensive testing on Atlas of the pattern of operand accesses. A high percentage of accesses is to a limited number of 'Named quantities' and the provision of a small associative buffer memory incorporating a 'Stack' facility is able to significantly reduce references to the main store and hence improve performance. The third concept involves the use of data descriptors which extend the flexibility of operand accesses by defining elements of a data structure which can be variable in length or arranged in the form of a string. The final feature is the connection of both the processor and its associated main store to a communication highway which links them to other units such as the Mass Core system, Disc backing store or even a second computer system.

In the detailed implementation of the complete system further use of associative buffer storage is used to minimize the effect of jump orders on instruction accesses and to readdress the storage units provided to give an element of 'fail soft' in that area.

Computer Structures: Past, Present and Future (abstract)

by ALAN KAY

Stanford University
Stanford, California

Computer Design in the seventies will be delineated by a number of past ghosts and present spectres.

The first is that IC manufacturers are highly motivated toward producing better "FORTRAN" components (faster linear addressed memories, adders, etc.) and thus most revolutionary designs will run FORTRAN more cost-effectively than the system for which they were intended. A traditional counter-example to this statement is the magic which can be done with CAMs. Unfortunately, the advent of cheap buffer storage largely obviates all but a masked search in terms of speed and the necessity to load the CAM still seems to be a serious liability for overall utility.

A second annoyance is also caused by the cheap buffer storage. It means that the effective memory (cache) cycle time is usually between 10 to 20 gate delays, which means that very little decoding can be done before a storage cycle is missed. Pipelining helps

alleviate this problem a bit but is antagonistic to highly branched or recursive evaluators. This means that it is difficult to hide the data paths on a machine when attempting to emulate with microcode (which in fact, now becomes quite "visible" itself).

A more serious problem to confront computer designers is the conspicuous absence of a new, more useful theory of evaluation on which to base a revolutionary design. There is some reason to believe that one such will appear in a few years, but for now, consequential processes which require essentially a B5500 environment are still being reinvented and understood after 10 to 15 years of life.

The social problems of users and customers who are unable to adapt to the cost/size tradeoffs implied by new technology seem to be part of the general inability to transcend "McLuhanism." The Grosch theory of the utility of the large processor (etc.) has not redeemed itself with proof. A "super" processor may give 10 to

20 improvement in speed; the elimination of secondary storage would improve many real jobs running in a real environment by 200 to 1000!

Mini's have embarrassed the computer establishment by being very cost effective compared to the large (747?) machines. Besides being more reliable their systems are not as bothered by the exponentially growing complexity problems involved in resource sharing that the dinosaurs face.

The above implies the following to this discussant: it is now quite possible to give most users their own (mini) processor, some memory and a link to various file systems. Computer "utilities" (as with power) will rent a service to handle global needs. The increase in actual computing power to a user may be substantial enough to allow him a few years of blessed peace from "Improvitis" during which time he will finally be able to invent a new theory of algorithmic computation which is so desperately needed.

A panel session—Computers in sports

The User's Reaction to Football Play Analysis and Player Ranking . . . Do They Make Any Difference?

by GIL BRANDT

Dallas Cowboys
Dallas, Texas

The Dallas Cowboys began to examine systematic ranking of college football players in the early 1960s using a computer.

These rankings are employed in the common NFL draft held in the early part of the year, after the Superbowl. When the draft is in progress, there is not a great deal of time available to make a selection. This means the player selection rankings must be available and provide a listing of the college players in an order most advantageous to the club. The computer was introduced into this ranking process to help eliminate many of the biases which existed previously.

In the beginning, many coaches and members of the team management were skeptical of the computer produced rankings. However, time has been in the favor of the computer, and as a result, most of the NFL teams now employ computer processing of the scouting information. The Dallas Cowboys stand behind their ranking system 100 percent. It has gained Dallas relatively unknowns, such as Calvin Hill, who was Rookie of the Year in 1969. The advantages should be obvious.

The computerized method of scouting has also brought some interesting side effects into the overall scouting process. There is now a standard and comprehensive form for all scouts to use. This is read directly by the computer. Scouts are now assigned to a particular area or region, and it has become less necessary to have prospective athletes scouted by many scouts (although the more the better) since the computer program has built-in weighting factors on the scouts themselves.

The Dallas Cowboys coaching staff also utilizes a football play analysis system on a weekly basis. The main advantage of this computer system is to make available analysis of opponent's plays much earlier in

the week (typically on Monday after a Sunday game). The tendencies which are found can be incorporated into the practice scrimmage sessions to get the offense and defense teams familiar with the type of attack used by the next weekend's opponent.

The computer is here to stay in professional football especially with the Cowboys. It was slow starting, but the coaches and team management are finally realizing it is an indispensable tool to aid in player selection and determine tendencies in an opponent's (or ones own) play sequence.

Computers and Scoreboards

by KEN EPPELE

Datex Division
Conrac Corporation

Computer capability in scoreboard control offers several features:

1. Data presentation in real time.
2. High speed timing data may be gathered, processed and presented to the fans in familiar units: miles per hour, time behind, etc.
3. Instantaneous comparison to statistical records.
4. Automatic message formatting.
5. High speed presentation, blinking, reversing.
6. Message recall.
7. Character generation including variable size characters, which aid flexibility and interest to message presentation.
8. High speed switching concepts permit animation and slide presentation.

Conrac Corporation has been applying computers to scoreboard display systems since 1967, when it delivered a mobile golf trailer to IBM, which has been used on the PGA golf circuit.

This was followed by the Oakland scoreboard which was the worlds first computer controlled electronic scoreboard to be installed in any major stadium. The computer was programmed to follow the play of the baseball game and develop up to the minute statistics during the course of the game. Each batter's average, for example, is shown when he comes up to bat and reflects his season to date average as of the last time at bat. The computer is also programmed so that a single entry can cause many events to occur. If, for example, the count is 3 and 2 on the batter with 2 outs, and the batter takes a third strike, the operator simply enters the strike code on the computer keyboard. Internally the computer updates the player and the team statistics, charging the batter with the strike-out, crediting the pitcher with one. The computer also causes the ball, strike, and out indications on the scoreboard to return to 0. It then updates the line score for the team just retired and automatically brings up the name and current average of the next batter.

At Ontario Motor Speedway, Conrac installed the worlds first automatic timing, scoring and display system for automobile racing. Radio transmitters, each generating a unique frequency, are mounted on each car in the field. Antennas buried in the track sense these signals and time any car that crosses an antenna. Time is measured to \pm one millisecond, which represents a distance of four inches at 200 miles per hour. This time data is processed by the computer and race order information is immediately displayed on in-field pylons which show the laps completed and the first nine car positions. During the race, computer print-outs make available more information, e.g. current order of the entire field, any car's fastest lap, average speed, and speed of the current lap, etc. Print-outs are distributed to the press and track announcers.

The scoreboard for the Dallas Cowboys incorporates video data terminals. The computer is programmed to maintain current team and player statistics during the play of the game. At any time during the game, preformatted messages may be recalled by the operator with the up to the minute statistics automatically included in the message. Messages may be previewed by the scoreboard director and displayed on the board at his direction.

The scoreboard system for the Stadium at Munich incorporates a minicomputer for primary control of the matrix scoreboard. This computer also communicates with a Siemens central computational center which includes a large data bank for all of the Olympic sports. This establishes a focal point for dissemination of information not only to the facility conducting the event, but to the facilities for other events. The Conrac com-

puter system accepts these types of messages, stores them temporarily and presents them at the main Stadium under command of the local operator. Conrac is also supplying the worlds first Mobile Matrix Scoreboard system to be used for display at the Canoeing and Regatta events. The mobile equipment is designed such that it may be operated at one remote facility one day and at another remote facility the next.

Computers have become an integral part of major scoreboard facilities and are virtually essential if they do nothing more than control the hardware in a flexible manner. Programming flexibility and ease of expansion allows for computer controlled scoreboards to accomplish other functions, such as statistical analysis or comparison and data processing, in real time during the course of the sporting event. Conrac foresees continued use of computers in future scoreboard installations, with even greater emphasis in the real time data processing of the sport.

Football Player Ranking Systems

by ATAM LALCHANDANI

Optimum Systems Incorporated
Palo Alto, California

The TROIKA player selection system has been the first breakthrough in the application techniques in the application of computer and statistical techniques in the area of personnel ranking in sports. The success of such a system can be evidenced by the fact that 23 of the 26 professional football clubs use a computerized system for guidance at the yearly draft meetings.

Based on some of the concepts researched above, extensive work has been done in developing a generalized system for ranking personnel in areas distinct from sports. It is felt that the tool developed herein can be a valuable aid in the decision-making concerned with the hiring and promotion of employees in government and public institutions. Technological advances can be made in defining jobs and personnel and the resultant optimum matching of the two.

OSI is in the process of marketing these ideas to industry and government and the next couple of years will show some concrete results in these areas. Currently, OSI sees itself in the research and development

stage, but it will not be long before we have an operational tool that would be a useful addition to all levels of management.

Portable Computer Timing, Analysis, and Display Systems for Rowing Applications; Legal Problems in Computer Sport Systems

by KENT MITCHELL

JAMCO, Inc.
Palo Alto, California

The firm JAMCO, Inc. has concentrated on making some of the lesser known sports more popular, using electronic timing devices and display systems. The computer has been used principally (1) to store and retrieve historical information about the sport and its competitors, (2) to analyze data generated by automatic and semiautomatic interval timing equipment, and (3) to operate electronic display equipment as a visual aid to spectators.

JAMCO's specific objective in rowing is to overcome certain spectator and press information problems, which are:

1. The inability presently to view the entire $1\frac{1}{4}$ mile long race from start to finish.
2. The lack of knowledge about the personal and competitive backgrounds of the 400 to 500 oarsmen who compete in major world-class regattas each year.
3. The poor press coverage given rowing for the above reasons and because typical "official timing" systems and methods for reporting results are inaccurate, unimaginative, misleading, and antiquated.

There has also been recent interest in computer systems which aid in competitive strategy and, as a result, attempt to predict the outcome of future competitions. Legal problems begin to appear when these techniques can actually affect the outcome.

Sports governing bodies, professional and amateur, have already begun to consider these problems, and some regulation has resulted. The implementation of real-time analysis and display systems during compe-

tion appears to be the area where the most regulation will be necessary. Additional restrictions may be imposed when these electronic devices used to generate data are attached to the equipment.

Computers in Track

by J. G. PURDY

TRW Incorporated
Sunnyvale, California

There have been a number of scoring tables which have been developed for track and field. The purpose of these tables is to compare performances (via a point score) between the different events, a higher score indicating a better performance. Historically, the scoring table was introduced as a necessity for the 1912 Olympic Games where the first decathlon was staged. Here, the scoring table provided a method to evaluate the best overall athlete in the 10 events of the competition.

The official ruling body in track athletics, the International Amateur Athletic Federation (IAAF), has adopted scoring systems in 1912, 1934, 1952, and 1962. Each succeeding scoring system was, supposedly, better than the previous system. Rule changes, new equipment and training methods greatly improved the performances in some events while leaving others behind; this destroyed the "equality" of the point score. The newer scoring tables attempted to reestablish the equality of the point scores and often were based on different principles.

The time has come again for the reevaluation of the current scoring system. New records are being made which causes some inequality in the point scores. However, the present tables were based on the physics of the performances only; physiological considerations were purposely ignored since the creators thought a fair system could be developed which was based on physics alone.

My current research has attempted to model the physiological effort associated with a performance rather than just the physics. A rather sophisticated computer program has been written to generate the scoring tables for the different events in the many various required formats. It is anticipated that these new tables will be presented to the IAAF for possible ratification as the official decathlon scoring tables.

It should be pointed out that the computer is almost essential in this task. Analysis of the thousands of performances and listing of the tables in the many different formats would almost be impossible without a computer.

Prospects for Sophistication in Football Play Analysis

by FRANK B. RYAN

U.S. House of Representatives
Washington, D.C.

It is a familiar task in football coaching ranks to analyze opponents from the point of view of trends and of statistical frequencies which might lead to a more confident decision on how to conduct game strategy. This analysis pervades the minds of football coaches at all levels but finds its fullest expression in the professional area.

To put things in perspective, a review of current techniques is worthwhile. Computer-aided analysis is not new to professional football and had its beginnings some years ago when modern methods confronted the age-old problem of assimilating a large quantity of data efficiently and effectively. The usual procedure begins by encoding football information which is then converted to machine-compatible form. Reports analyzing information covering several games are generated in a pre-structured format which invariably is finalized once and for all prior to the beginning of a season. Generally these reports produce simple frequency counts for a variety of situations and leave interpretation and weighting of results up to the coach. The really useful

advantages of computer assistance apparently are only faintly recognized by most modern-day coaching staffs, including those who claim a heavy dependence upon this tool.

There are many reasons for this current imbalance. Tradition, of course, plays an important role as well as the very human element of "teaching an old dog new tricks." In addition, many coaches believe that a simple approach must be the best approach. The main force here appears to be severe changes which computer aids impose on the mode of coaching operation. And yet these changes have not been supported fully by theoretical developments which would inspire confidence in the new methods.

As a first step in attacking the general problems facing computer-aided strategy analysis today, a general retrieval system, called PROBE, was conceived. The plan was to develop a system with great flexibility in three important areas: data base definition, report generation, and formatting of the output display. This system caters to the individual interests and methods of different coaching staffs and attempts to bring computer assistance to the coach, rather than the coach to the computer. At best this approach is intermediate in the view of providing modern technological aids to football strategy analysis.

Looking ahead to the future, one wonders if the game of football can survive the impact of sophistication. There certainly is a point of diminishing returns which, however, has not yet been reached, and ultimately the flavor of professional football would appear to be in jeopardy from excessive analysis. To provide an adequate base for a sophistication in game strategy which does not compromise the game's basic appeal, there needs to be attention devoted to a number of areas, among them linguistics, decision theory, and the interplay between team, mass, and individual psychologies. Interestingly enough, the game does and will continue to provide a fruitful model for fundamental studies in each of these areas.

On the hybrid computer solution of partial differential equations with two spatial dimensions*

by GEORGE A. BEKEY

University of Southern California
Los Angeles, California

and

MAN T. UNG

Dillingham Environmental Company
La Jolla, California

INTRODUCTION

For a number of years it has been suggested that one of the fruitful areas of application for hybrid computation might lie in the study of distributed parameter systems.^{1,2} In principle, the combination of analog computer speed with the memory and logical capabilities of digital machines should make it possible to solve partial differential equations both efficiently and rapidly. However, most of the published work in the field has dealt only with linear problems in one spatial dimension and time, where the advantages and limitations of the particular methods do not stand out clearly.

The purpose of this paper is to present a detailed analysis of three methods applicable to the hybrid solution of nonlinear parabolic partial differential equations with two spatial dimensions, and to apply two of these methods to a specific "benchmark" problem. By using different methods to solve the same problem, their relative merit can be evaluated in proper perspective.

REVIEW OF HYBRID METHODS FOR ONE-DIMENSIONAL PARTIAL DIFFERENTIAL EQUATIONS

As a background to the study of two-dimensional systems, the major techniques for one-dimensional equations will be reviewed briefly. Consider a linear,

* This research was supported in part by the U.S. Air Force Office of Scientific Research under Grant Number AFOSR 71-2008.

one-dimensional diffusion equation

$$a \frac{\partial^2 U(x, t)}{\partial x^2} = \frac{\partial U(x, t)}{\partial t} \quad 0 \leq x \leq L \quad (1)$$

with the following Dirichlet boundary conditions

$$\begin{aligned} U(0, t) &= g(t) \\ U(L, t) &= c(t) \end{aligned} \quad (2)$$

and the initial condition

$$U(x, 0) = b(x) \quad (3)$$

Various alternative hybrid computer methods for solution of (1) differ primarily in the way in which the variables x or t or both are discretized, in order to obtain ordinary differential equations or algebraic equations.

Discrete-space-continuous-time approximations

In this method the x -domain is represented by discrete stations or nodes which divide the x -axis into M segments.* The nodes are numbered consecutively from 0 to M and Equation (1) is approximated by the system of ordinary differential equations

$$\begin{aligned} \frac{dU_i(t)}{dt} &= \frac{a}{(\Delta x)^2} [U_{i+1}(t) - 2U_i(t) + U_{i-1}(t)] \quad (4) \\ U_i(0) &= b_i; \quad i = 1, 2, 3, \dots, M-1 \end{aligned}$$

* The segments are generally of equal length, but need not be. For example, in the semi-infinite domain $0 \leq x < \infty$ it may be more convenient to divide the x -axis using a logarithmic or exponential scale.

where, by convention, $U_i(t)$ denotes $U(i\Delta x, t)$ and b_i denotes $b(i\Delta x)$. If the number of stations M is not too large one can solve Equation (4) in parallel, entirely on the analog computer.

Continuous-space-discrete-time approximation

In this method the spatial variable x is kept continuous while the temporal variable t is represented by a sequence of discrete steps, $n\Delta t$, $n=0, 1, 2, \dots, P$. Then a finite-difference approximation to Equation (1) may be written as

$$\frac{d^2 U_n(x)}{dx^2} = \frac{U_n(x) - U_{n-1}(x)}{a\Delta t} \quad (5)$$

where, by definition

$$\begin{aligned} U_n(x) &= U(x, n\Delta t) \\ U_n(0) &= g_n \\ U_n(L) &= c_n \end{aligned}$$

This is a split-boundary-value problem whose solution consists of hyperbolic functions at each time step. It can be shown^{3,4} that direct solution of (5) by iterating the unknown initial condition $dU_n(0)/dt$ until the second boundary is matched leads to computational instability.

Method of decomposition

In order to avoid the stability problems inherent in the classical solution, Vichnevetsky⁵ introduced his Method of Decomposition which amounts to breaking Equation (5) into two stable first-order differential equations. One is integrated in the forward direction while the other along the backward direction.

Another approach to the stability problem in Equation (5) was used by Hara⁶ who transformed the problem into one of optimal control.

Monte Carlo methods

Finally, Monte Carlo Methods^{7,8} have been used (in two and three spatial dimensions) for the evaluation of the variable U at a limited number of points in the space under consideration.

The application of hybrid computers to the solution of two-dimensional problems is based on generalization of the first three methods discussed above. Three methods have been used successfully in solving two-dimensional partial differential equations: the "Component-Sharing Method," the "Explicit/Implicit

Method" and the hybrid implementation of the Alternating-Direction Implicit procedure.

THE "COMPONENT SHARING" METHOD

This method is a generalization of the discrete-space-continuous-time procedure because only the spatial dimensions are discretized. The Component-Sharing method is designed to reduce the number of analog components required to solve all stations of the $x-y$ plane in parallel. The idea is to divide the net into equal subdivisions so that there is enough analog equipment to simulate every node within one subdivision in parallel. Then the same analog circuits are used to simulate other subdivisions in a certain order until the whole net is covered. At each station we obtain an approximation $U_{i,j}^k(t)$ to the true solution. The superscript k refers to the iteration number. The procedure is to iterate across the medium again, solving each subdivision serially, in the same order as designated above, to come up with a better approximation $U_{i,j}^{k+1}(t)$ based upon the initial and boundary conditions. In subsequent iterations, the values $U_{i,j}(t)$ are the ones obtained from the previous computation. In this method the digital computer's role is that of control plus data storage and playback. The first attempt to use this method was carried out by Howe and Hsu⁹ on the IBM 7090 using a fourth-order Runge-Kutta formula to simulate the analog computer integration.

Consider the following linear parabolic equation:

$$a \left[\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} \right] = \frac{\partial U}{\partial t} \quad 0 \leq x, y \leq L \quad (6)$$

with boundary and initial conditions

$$\begin{aligned} U(0, y, t) &= g(y, t) \\ U(L, y, t) &= c(y, t) \\ U(x, 0, t) &= h(x, t) \\ U(x, L, t) &= r(x, t) \\ U(x, y, 0) &= b(x, y) \end{aligned} \quad (7)$$

There are many ways of creating a net. For example, the shape may be square, rectangle, an entire row or an entire column. The last alternative is adopted in this paper. Allowing t to be the analog independent variable we end up with the following set of difference-differential equations representing (6) for $a=1$:

$$\begin{aligned} dU_{i,j}^k(t)/dt &= \mu U_{i+1,j}^{k-1}(t) + \mu U_{i-1,j}^k(t) + \lambda U_{i,j+1}^k(t) \\ &+ \lambda U_{i,j-1}^k(t) - 2(\mu + \lambda) U_{i,j}^k(t) \end{aligned} \quad (8)$$

for

$$i = 1, 2, 3, \dots, M-1$$

$$j = 1, 2, 3, \dots, N-1$$

where μ and λ are defined as

$$\mu = 1/(\Delta x)^2$$

$$\lambda = 1/(\Delta y)^2$$

Equation (8) indicates that we need one integrator per node on the $i-j$ plane. If we have enough integrators, the whole column can be simulated on the analog computer, the same equipment being used sequentially to simulate columns $i=1, 2, 3, \dots, M-1$ in that order. During the first iteration, solutions for all but the first column must be assumed to provide a set of starting boundary conditions for each column. Each new iteration yields a closer approximation to the true solution to Equation (6). The solutions are said to be convergent whenever, for an arbitrary $\epsilon > 0$, there exists a number K such that for all $k > K$ and for all i, j and n

$$| U_{i,j}^k(n\Delta t) - U_{i,j}^{k-1}(n\Delta t) | < \epsilon \quad (9)$$

In practice, the computation is stopped as soon as, for a chosen ϵ , Equation (9) is satisfied for the first time. The boundary conditions find their way into Equation (8) at all interior points adjacent to the edges.

It can be shown that the solution is unconditionally stable.¹⁵

Possible improvements

It is feasible to save memory space or shorten the execution time or both if a slightly different approach is taken. This is necessary because the simulation of one column at a time, as described, possesses some limitations. For example, we find it prohibitive to implement the method if $j=1, 2, 3, \dots, 200$. The need for a better plan leads us to working with a rectangular block of nodes instead of a whole column. The cluster shown in Figure 1 accounts for $I \times J$ points but we need to use only one converter channel per outside node (marked by asterisks), or a total of $[2I + 2(J-1)]$ channels. The potential of this scheme is fully realized at large values of I and J . For square blocks, we observe that while the number of nodes (J^2) grows quadratically with J , the number of converter channels required increases linearly, $4(J-1)$. Thus a 10×10 block calls for only 36 DAC channels. Because the time histories of the interior nodes are not stored, the memory requirement drops to 36 percent of the total storage space when all interior points are to be kept in the

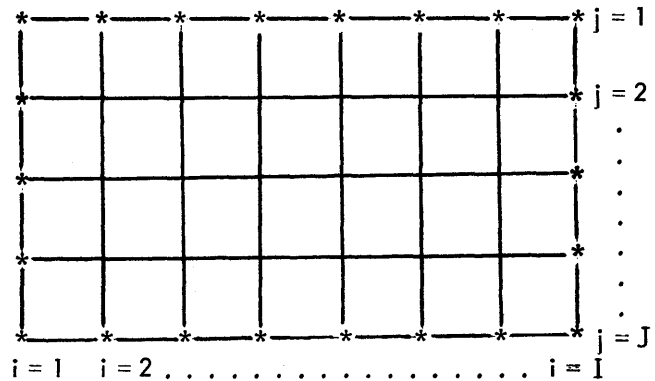


Figure 1—A subdivision of the X-Y plane

digital computer. From a macro point of view, the execution time is also cut down to approximately 36 percent compared to the simulation of a single column at a time. We should reiterate that even though only 36 percent of the converters are utilized the run time is not reduced to exactly 36 percent due to a fixed amount of overhead in the program. On top of this, some time is spent in initializing the interior points during the RESET cycle. The same DACs could be used for establishing initial conditions.

As described herein, the integration with respect to time was carried out continuously for $0 \leq t \leq t_{max}$. If t_{max} is of long duration, the Component-Sharing method could be beneficially applied to a fraction of the time t_{max} , say $0 \leq t \leq t_{max}/p$, where p is an integer. After the solution converges for this interval we then move on and solve for $U(x, y, t)$, $t_{max}/p < t \leq 2t_{max}/p$, and so on. There should be an optimal p in terms of solution speed and accuracy for each problem.

HYBRID IMPLEMENTATION OF THE ALTERNATING DIRECTION METHOD

The Alternating Direction Implicit method was suggested originally by Douglas¹⁰ and Peaceman and Rachford,¹¹ and its digital computer implementation discussed in numerous papers.¹⁶⁻¹⁹ Recently Bishop¹² demonstrated successfully an adaptation of the algorithm to hybrid computation. The basic Alternating Direction Implicit method can be used to solve Equation (6) as follows: let the coefficient $a=1$, and define

$$U_{i,j}^* = U(i\Delta x, j\Delta y, (n + \frac{1}{2})\Delta t) \quad (10)$$

Then we can approximate Equation (6) by two difference equations which are used in turn, over successive

time steps, each of duration $\Delta t/2$:

$$\frac{U_{i,j}^* - U_{i,j,n}}{(\Delta t/2)} = \delta_x^2 [U_{i,j}^*] + \delta_y^2 [U_{i,j,n}] \quad (11)$$

for $i = 1, 2, \dots, M-1$

$$\frac{U_{i,j,n+1} - U_{i,j}^*}{(\Delta t/2)} = \delta_x^2 [U_{i,j}^*] + \delta_y^2 [U_{i,j,n+1}] \quad (12)$$

for $j = 1, 2, \dots, N-1$

Equations (11) and (12) are implicit in x and y respectively, resulting in systems of algebraic equations which can be solved using matrix inversion.

By adapting Equations (11) and (12) to the hybrid computer we can entirely avoid matrix manipulation which accounts for the greater part of the time spent in solving the problem digitally. In order to simplify the analog circuits associated with this algorithm let us define, for $n\Delta t \leq t \leq (n+1)\Delta t$

$$\left. \begin{aligned} \hat{u}_{i,j}(t) &= U_{i,j}(t) - U_{i,j}(n\Delta t) \\ u_{i,j}(t) &= U_{i,j}(t) - U_{i,j}((n+\frac{1}{2})\Delta t) \\ \pi_{i,j}(n\Delta t) &= U_{i+1,j}(n\Delta t) + U_{i-1,j}(n\Delta t) \\ &\quad + U_{i,j+1}(n\Delta t) + U_{i,j-1}(n\Delta t) \\ &\quad - 4U_{i,j}(n\Delta t) \end{aligned} \right\} (13)$$

Keeping the independent variable t continuous while discretizing x and y ($\Delta x = \Delta y$) we can approximate Equation (6) as follows

$$\begin{aligned} d\hat{u}_{i,j}(t)/dt &= [(\Delta x)^2]^{-1} [\hat{u}_{i+1,j}(t) + \hat{u}_{i-1,j}(t) \\ &\quad + \hat{u}_{i,j+1}(t) + \hat{u}_{i,j-1}(t) - 4\hat{u}_{i,j}(t) \\ &\quad + \pi_{i,j}(n\Delta t)] \end{aligned} \quad (14)$$

The hybrid computer implementation of the Alternating-Direction Implicit method revolves around two important steps designed to advance the time dimension t by $\frac{1}{2}\Delta t$ per step.

Step 1: In this step we perform calculations that are implicit in the x direction. Let us modify Equation (14) to attain this goal

$$\begin{aligned} d\hat{u}_{i,j}(t)/dt &= [(\Delta x)^2]^{-1} [\hat{u}_{i+1,j}(t) - 2\hat{u}_{i,j}(t) \hat{u}_{i-1,j}(t) \\ &\quad + \pi_{i,j}(n\Delta t)] + [(\Delta x)^2]^{-1} [\hat{u}_{i,j+1}(n\Delta t) \\ &\quad - 2\hat{u}_{i,j}(n\Delta t) + \hat{u}_{i,j-1}(n\Delta t)] \end{aligned} \quad (15)$$

Every term inside the second square bracket of Equation (15) is zero by virtue of the definition on Equation (13). Using a system of coupled analog circuits we

solve for $\hat{u}_{i,j}(t)$

$$\begin{aligned} \hat{u}_{i,j}(t) &= [(\Delta x)^2]^{-1} \int_{n\Delta t}^{(n+1/2)\Delta t} [\hat{u}_{i+1,j}(t) - 2\hat{u}_{i,j}(t) \\ &\quad + \hat{u}_{i-1,j}(t) + \pi_{i,j}(n\Delta t)] dt \quad \text{for} \\ &\quad i, j = 1, 2, 3, \dots, M-1 \end{aligned} \quad (16)$$

The system of equations depicted by (16) results in $M-1$ coupled analog circuits. Each analog circuit corresponds to a subscript i . The quantities $\pi_{i,j}(n\Delta t)$ are forcing functions coming from the digital machine. The integration of (16) must be carried out $M-1$ times, corresponding to $M-1$ possible values of the subscript j . In other words, subscript i is taken care of in parallel while subscript j is solved serially. Step 1 advances the time dimension by $\frac{1}{2}\Delta t$, from $n\Delta t$ to $(n+\frac{1}{2})\Delta t$.

Step 2: Starting with equation (14) again, we modify it to read

$$\begin{aligned} du_{i,j}(t)/dt &= [(\Delta x)^2]^{-1} [u_{i,j+1}(t) - 2u_{i,j}(t) \\ &\quad + u_{i,j-1}(t) + \pi_{i,j}((n+\frac{1}{2})\Delta t)] \\ &\quad + [(\Delta x)^2]^{-1} [u_{i+1,j}((n+\frac{1}{2})\Delta t) \\ &\quad - 2u_{i,j}((n+\frac{1}{2})\Delta t) + u_{i-1,j}((n+\frac{1}{2})\Delta t)] \end{aligned} \quad (17)$$

Similar reasoning to step 1 shows that the second half of the right-hand-side of Equation (17) is null. Integrate (17) to obtain $u_{i,j}(t)$

$$\begin{aligned} u_{i,j}(t) &= [(\Delta x)^2]^{-1} \int_{(n+1/2)\Delta t}^{(n+1)\Delta t} [u_{i,j+1}(t) - 2u_{i,j}(t) \\ &\quad + u_{i,j-1}(t) + \pi_{i,j}((n+\frac{1}{2})\Delta t)] dt \end{aligned} \quad (18)$$

The same $M-1$ analog circuits used in step 1 are put to work again in step 2, except the roles of the subscripts i and j are reversed. Note that the computation in step 2 is implicit in y and explicit in x . Also note that the hybrid implementation of the Alternating-Direction Implicit method precludes matrix inversion. Since a great part of the time is devoted to matrix inversions in the digital solution, the prospect of significant improvement in the hybrid solution speed is potentially real. The change of variable from $U_{i,j}(t)$ to $\hat{u}_{i,j}(t)$ and $u_{i,j}(t)$ enables us to use the entire dynamic range of the analog integrators during each half time step, $\frac{1}{2}\Delta t$.

EXPLICIT-IMPLICIT METHOD

This method is a generalization of the "method of decomposition" mentioned earlier in connection with discrete-time-continuous space approximations. The integration time of the analog computer is equated to

one of the spatial dimensions, say x , while y and t are discretized, thus producing a set of second-order differential equations with split-boundary conditions whose solution grows without bound as x increases. The Explicit/Implicit Method leads to the decomposition of the unstable second-order differential equation at each (y, t) node into two stable first-order differential equations. One of these is to be integrated along the x -axis and the other along the $-x$ direction. To illustrate this point, let us approximate Equation (6) as follows

$$\frac{d^2 U_{j,n}}{dx^2} = - \frac{U_{j+1,n-1} - 2U_{j,n-1} + U_{j-1,n-1}}{(\Delta y)^2} + \frac{U_{j,n} - U_{j,n-1}}{\Delta t} \quad (19)$$

$$\frac{d^2 U_{j,n}}{dx^2} - \frac{1}{\Delta t} U_{j,n} = - \frac{1}{\Delta t} U_{j,n-1} - \frac{1}{(\Delta y)^2} \times [U_{j+1,n-1} - 2U_{j,n-1} + U_{j-1,n-1}] \quad (20)$$

Equation (20) is unstable if a direct approach is attempted to solve it. Simply reversing the direction of integration¹³ won't help because $dx^2 = d(-x)^2$. Rewriting Equation (20) by indicating the dependence of $U_{i,j}$ on x

$$[d^2 U_{j,n}(x)/dx^2] - \Psi U_{j,n}(x) = -\lambda U_{j+1,n-1}(x) - \nu U_{j,n-1}(x) - \lambda U_{j-1,n-1}(x) \quad (21)$$

where

$$\Psi = \frac{1}{\Delta t}$$

$$\nu = \frac{1}{\Delta t} - \frac{2}{(\Delta y)^2}$$

All the terms on the right-hand-side of Equation (21) are known because they belong to the last time plane, $(n-1)\Delta t$. Let us denote them by $R_j(x)$. The solution of Equation (21) is thus explicit with respect to time. Dropping the temporal subscripts n whenever we imply the $n\Delta t$ time plane, Equation (21) becomes

$$[d^2 U_j(x)/dx^2] - \Psi U_j(x) = R_j(x) \quad (22)$$

In order to solve this equation we proceed as follows. First, let us solve the first-order ordinary differential equation

$$dV_j(x)/dx + (\Psi)^{1/2} V_j(x) = R_j(x) \quad 0 \leq x \leq 1 \quad (23)$$

This is a stable equation if it is integrated in the $+x$ or forward direction. Secondly, define an equation that

is stable in the backward $(-x)$ direction

$$dZ_j(x)/dx - (\Psi)^{1/2} Z_j(x) = V_j(x) \quad 0 \leq x \leq 1 \quad (24)$$

i.e., the independent variable x decreases from 1 to 0 during each run. Instead of working on Equation (24) directly we solve the following expression

$$dZ_j(x)/d(-x) - (\Psi)^{1/2} Z_j(x) = V_j(-x) \quad (25)$$

which is also integrated in the forward direction. Then

$$d^2 Z_j(x)/dx^2 - \Psi Z_j(x) = [d/dx + (\Psi)^{1/2}][d/dx - (\Psi)^{1/2}] Z_j(x) = [d/dx + (\Psi)^{1/2}] V_j(x)$$

or

$$d^2 Z_j(x)/dx^2 - \Psi Z_j(x) = R_j(x) \quad (26)$$

Hence $Z_j(x)$ satisfies the ordinary differential Equation (22) but not necessarily the boundary conditions. This is true for any initial values $V_j(0)$ and $Z_j(M)$. For simplicity it is easy to choose $V(0, j) = 0$ and $Z(M, j) = 0$; also $M = x_{\max} = 1$.

Define two homogeneous ordinary differential equations

$$\left. \begin{aligned} dW1(x)/dx + (\Psi)^{1/2} W1(x) &= 0 \\ dW2(x)/dx - (\Psi)^{1/2} W2(x) &= 0 \end{aligned} \right\} 0 \leq x \leq 1 \quad (27)$$

with the following initial values

$$W1(0) = W2(M) = 1$$

The equation for $W1(x)$ is to be integrated in the forward $(+x)$ direction while that for $W2(x)$ in the backward direction (from $x=1$ to $x=0$). Also let $U_j(x)$ be composed of

$$U_j(x) = Z_j(x) + a_j W1(x) + b_j W2(x) \quad (28)$$

where a_j and b_j are two constants to be evaluated later. According to Equations (26) and (27) this equality becomes

$$(d^2/dx^2 - \Psi) U_j(x) = R_j(x) \quad (29)$$

Equation (29) indicates that the right-hand-side of (28) satisfies the ordinary differential Equation (22). The same right-hand-side of (28) can satisfy the boundary conditions $U_{j,n}(0)$ and $U_{j,n}(M)$ too, if we set

$$U(0, j, n) = Z(0, j) + a_j W1(0) + b_j W2(0)$$

$$U(M, j, n) = Z(M, j) + a_j W1(M) + b_j W2(M) \quad (30)$$

Equation (30) contains two unknowns, a_j and b_j , which can be readily determined. With a_j and b_j solved, $U_j(x)$ can be directly computed from the combination of $Z_j(x)$, $W1(x)$ and $W2(x)$ using Equation (28).

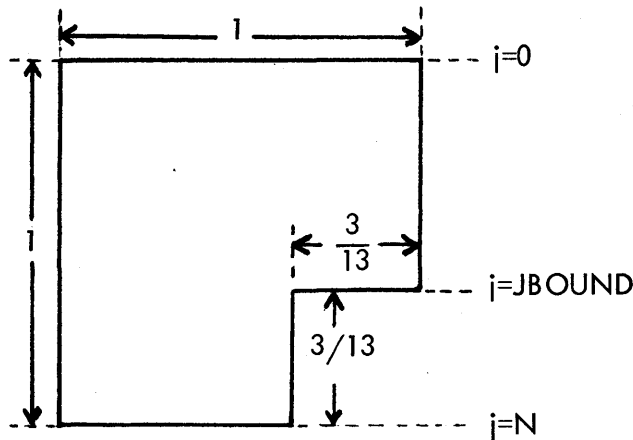


Figure 2—Cross-section of the bar under consideration

NONLINEAR TEST PROBLEM

In order to compare the various methods, we have applied them to the solution of a nonlinear heat conduction problem in a bar with the irregular cross-section shown in Figure 2, and described by

$$a(U)[\partial^2 U / \partial x^2 + \partial^2 U / \partial y^2] = \partial U / \partial t$$

$$0 \leq x, y \leq 1, t \geq 0 \quad (31)$$

with U on the boundaries being uniformly at unity and with initial condition

$$U(x, y, 0) = 1 - \sin \pi x \cdot \sin \pi y \quad (32)$$

Note that the initial temperature has a jump discontinuity along the cutout of the cross-section. The thermal diffusivity is assumed to be given by

$$a(U) = .8 + .2U$$

This problem is set up so as to have a temperature profile symmetrical about the main diagonal of the cross-section, in order to provide a convenient check on error propagation along the x -axis in various methods. We will investigate the problem for $0 \leq t \leq .08$ second. During this transient period most errors are expected to occur. Also during this period the temperatures at most of the points reach 90 percent of their steady-state values.

Component-sharing method

This method leads to the system of difference-differential equations

$$dU_{i,j} / dt = a(U_{i,j}) [\mu U_{i+1,j} + \mu U_{i-1,j} + \lambda U_{i,j+1} + \lambda U_{i,j-1} - 2(\mu + \lambda) U_{i,j}] \quad (33)$$

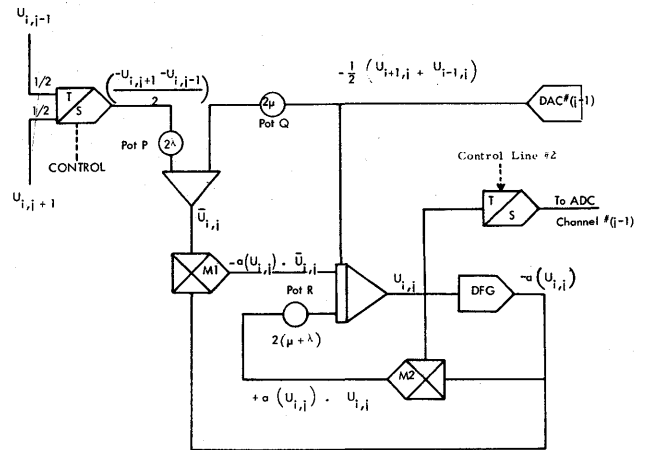


Figure 3—Component-sharing method for non-linear diffusion equation. In this figure

$$\hat{U}_{i,j} = [\mu U_{i+1,j} + \mu U_{i-1,j} + \lambda U_{i,j+1} + \lambda U_{i,j-1}]$$

CONTROL is a pulse train starting at the beginning of each frame time.

which are identical to Equation (8) except for the temperature-dependent term $a(U_{i,j})$. It can be restated to read

$$dU_{i,j} / dt = a(U_{i,j}) \cdot \hat{U}_{i,j} \quad (34)$$

where $\hat{U}_{i,j}$ represents the quantity between the brackets in (33). Equation (34) produces the analog diagram in Figure 3. The complete program consisted of 12 such building blocks corresponding to 14 points on the y -axis (the two exterior points are known boundaries).

Figure 4 depicts the flowchart of this program. We have found that large errors are introduced if we carelessly mix the discrete and the continuous signals to

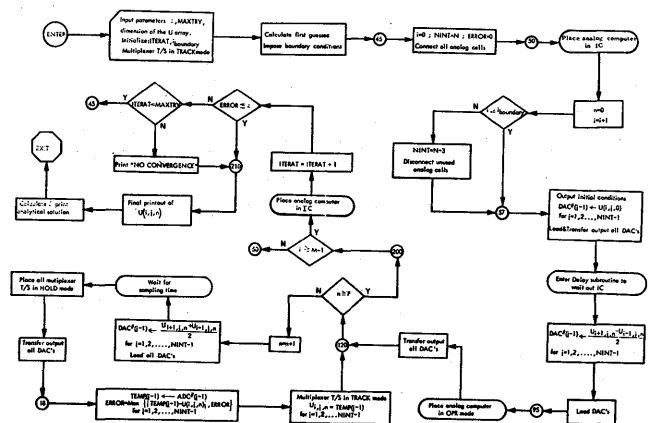


Figure 4—Flowchart for the method of component-sharing (Nonlinear diffusion equation)

form $\bar{U}_{i,j}$ in Figure 3. That is why the continuous analog coupling terms $U_{i,j-1}$ and $U_{i,j+1}$ are sampled before they are added to the output of the DAC. A storage block of $14 \times 14 \times 53$ words is needed for the data of the program. The digital portion of the hybrid program, as written, will work for any nonlinearity. Variations occur only in the analog program where different functions have to be set into the DFGs. Directing the information flow between the analog computer and the mass storage is the main task of the digital computer. In practice it is not feasible to write one hybrid program to fit every occasion because changes in the initial conditions and boundary conditions result in scaling problems. Digital computer users grow accustomed to the convenience of floating-point arithmetic. Hybrid programs, especially the analog portion, are restricted to a fixed-point regime. For example, Figure 3 should be rescaled if $\max U_{i,j}(t) < 0.5$. In that case, only a fraction of the dynamic range of the analog computer is fully utilized. As it stands, Figure 3 is not yet "optimally scaled" because $0 \leq U_{i,j}(t) \leq 1$. In other words, only one-half of the analog computer dynamic range is covered by the variable $U_{i,j}(t)$. We could have offset $U_{i,j}(t)$ in such a way that its scaled value reads

$$-1.0 \leq 2[U_{i,j}(t) - 0.5] \leq 1.0$$

This type of biasing of a variable is sometimes done when accuracy considerations override the extra expenditure of time and equipment. As the solution develops, the program types out the convergence error ϵ for each iteration to inform the operator of the progress being made. The quantity ϵ was defined in Equation (9) and in the current program

$$\epsilon = \max | U_{i,j,n}^k - U_{i,j,n}^{k-1} |$$

for all i, j and n . It was found that, starting at the same initial guesses, the nonlinear problem converges at the same rate as its linear counterpart ($a(U) = 1$). One possible reason is the fact that the chosen nonlinearity $a(U)$ does not vary drastically with the temperature U . A typical computer output shows that the solution converges to within .1 percent in 20 iterations.

The implicit alternating direction method

The application of this method to the benchmark problem is not available at this time. However, Bishop has a working program* on the performance of a gas storage reservoir, also a diffusion equation. He implemented a linear model of the reservoir in two spatial dimensions on the EAI 8900 hybrid computing system.

For a grid of 15×15 in the spatial dimensions and for 14 time steps his program consumed 2.6 seconds, and his results agreed with comparable digital results within 1 part per 10,000. It is safe to assume that the execution time would remain essentially at 2.6 seconds for the nonlinear problem as well, if one is willing to handle the nonlinearity on the analog computer. The situation in this case is similar to that of the Component-Sharing method. All we need is to add some DFGs to the existing analog diagram.

Solution of the test problem by the explicit/implicit method

Equation (31) becomes

$$d^2 U_{j,n}(x) / dx^2 = -\delta_v^2 [U_{j,n-1}(x)] + [U_{j,n}(x) - U_{j,n-1}(x)] / [a(U_{j,n}(x)) \cdot \Delta t] \quad (35)$$

The above expression can be put in the form similar to that of (22), but this time we have a system of nonlinear differential equations. For $j = 1, 2, 3, \dots, N-1$

$$d^2 U_{j,n}(x) / dx^2 - \Theta_j(U_{j,n}(x)) \cdot U_{j,n}(x) = R_j(x) \quad (36)$$

where

$$\Theta_j(U_{j,n}(x)) = [a(U_{j,n}(x)) \cdot \Delta t]^{-1} \quad (37)$$

$$R_j(x) = -\lambda U_{j+1,n-1}(x) - \nu_j(U_{j,n}(x)) U_{j,n-1}(x) - \lambda U_{j-1,n-1} \quad (38)$$

and

$$\nu_j(U_{j,n}(x)) = [a(U_{j,n}(x)) \cdot \Delta t]^{-1} - 2\lambda \quad (39)$$

The Explicit/Implicit method is based upon the superposition principle in the decomposition procedure and in matching the boundary conditions of $U_{j,n}(x)$. To apply this method we must linearize Equation (36). Linearization which yields a set of linear differential equations with varying coefficients, rests upon the search for a purely space-dependent coefficient $\Psi_j(x)$ such that $\Theta_j(U_{j,n}(x)) \simeq \Psi_j(x)$. Once a suitable function $\Psi_j(x)$ is found the Explicit/Implicit method can be applied to the solution of the equation

$$d^2 U_{j,n}(x) / dx^2 - \Psi_j(x) U_{j,n}(x) = R_j(x) \quad (40)$$

A predictor-corrector approach to this problem was suggested by Vichnevetsky.¹⁴

Let the "predicted value" $U_{j,n}^*(x)$ be represented by the linear sum

$$U_{j,n}^*(x) = \sum_{k=1}^K c_k U_{j,n-k}(x) \quad (41)$$

where c_k are constants. Substituting $U_{j,n}^*$ into the

* The information was supplied by Dr. Kenneth Bishop to the authors in a private communication.

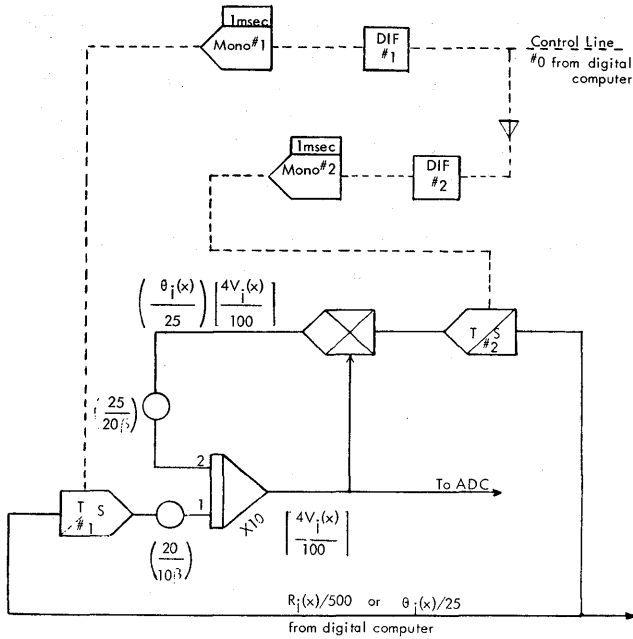


Figure 5—Explicit/implicit method analog diagram. Logic signals are shown by dashed lines. Mono is the abbreviation for monostable multivibrator

right-hand side of Equation (37) yields

$$\Psi_j(x) = [a(U_{j,n}^*(x)) \cdot \Delta t]^{-1} \quad (42)$$

A space-dependent term comparable to $v_j(U_{j,n}(x))$, say $v_j(x)$, must be found by using the same substitution as done in (42). With the help of $\Psi_j(x)$ and $v_j(x)$ we are prepared to solve for $U_{j,n}(x)$ in Equation (40). The newly found $U_{j,n}(x)$ can be considered as the “corrected” value. If $|U_{j,n}^*(x) - U_{j,n}(x)|$ is still deemed too large, we can always reinsert $U_{j,n}(x)$ into the coefficient (42) and again solve for yet another new value of $U_{j,n}(x)$. The process can be repeated a number of times to improve the approximation. Hence the original nonlinear Equation (36) can be approached as closely as desired.

Straightforward decomposition as done earlier is no longer permissible, since Ψ is now a function of x . To solve the nonlinear problem we define

$$d\theta_j(x)/dx + \theta_j^2(x) = \Psi_j(x) \quad (43)$$

This is a Riccati differential equation. It can be shown¹⁵ that its solution is well-behaved for the problem under consideration. We now apply the Explicit/Implicit method, using $\theta_j(x)$ instead of $\Psi_j(x)^{1/2}$, and obtain

$$dV_j(x)/dx + \theta_j(x) \cdot V_j(x) = R_j(x) \quad (44)$$

$$dZ_j(x)/dx - \theta_j(x) \cdot Z_j(x) = V_j(x) \quad (45)$$

It can be verified (by substitution) that $Z_j(x)$ actually satisfies the ordinary differential Equation (40).

Figure 5 contains the analog diagram for one cell of the problem. The flowchart can be found in Figure 6.

Results

Since an analytical solution to the problem is not known, a reference solution $\bar{U}_{i,j}(t)$ was generated digitally using the implicit alternating direction method. We then chose to evaluate the hybrid solution $U_{i,j}(t)$ by computing the criterion function

$$E_{i,j}(t) = |\bar{U}_{i,j}(t) - U_{i,j}(t)|$$

which is the absolute value of the difference between the digital and the hybrid solutions. The time histories of three points along the main diagonal of the $x-y$ plane are singled out for our study. Point (1, 1) is closest to the upper left corner (see Figure 2); point (6, 6) is right at the center of the $x-y$ plane and point (9, 9) on the main diagonal is closest to the cut-out. Figure 7 contains the error plots for these three points. It is evident that the accuracy of the two methods is of the same order of magnitude.

The solution time (using the EAI690 hybrid computer) were:

Component-sharing method—8 seconds

Explicit-implicit method —8 minutes

However, the latter time should not be taken as a norm for the explicit-implicit method. By allocating

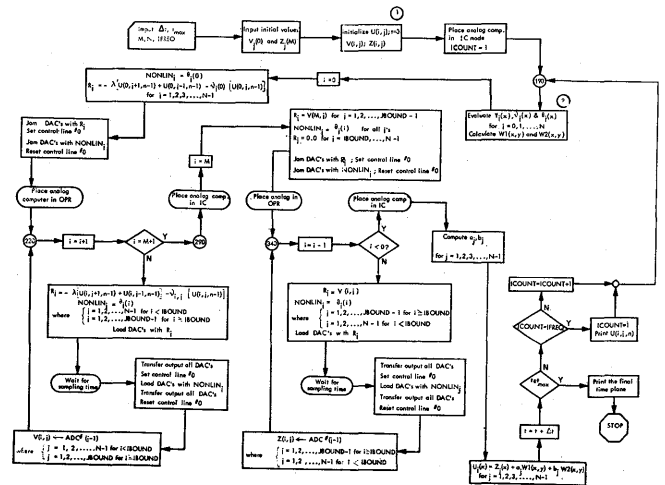


Figure 6—Explicit/implicit method flowchart

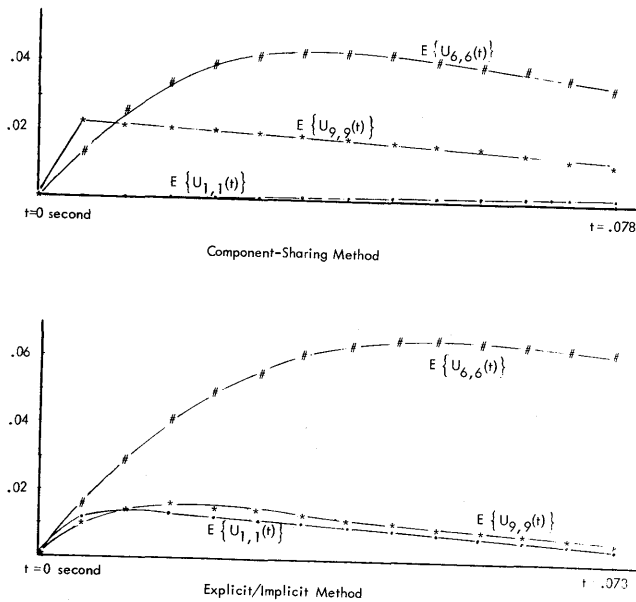


Figure 7—Errors in the nonlinear solutions. In this figure $E\{U_{i,j}(t)\} = |\text{Analytical } U_{i,j}(t) - \text{Computed } U_{i,j}(t)|$.

the solution of $W1(x, y)$ and $W2(x, y)$ to the analog computer, the 8 minutes can be reduced to approximately 10 seconds. The all-digital solution on the small fixed-point digital computer system required 38 minutes. Evidently, this time can be reduced drastically by using larger machines.

DISCUSSION OF RESULTS AND CONCLUSIONS

Based upon the experience encountered we draw the following conclusions about the merits of the two hybrid methods examined in detail in this paper:

Component-sharing method

1. Iterations are needed because the starting values for the interior nodes are not known at the beginning of the computation.
2. Solution is unfolded as a function of time.
3. Unconditionally stable.¹⁵
4. Solution speed is limited only by the digital computer.
5. A general executive program can be written to handle both linear and nonlinear equations, even to accommodate a class of geometrical configurations.

One severe drawback of this method is the large memory requirement for any practical problem.

Explicit/implicit method

1. No iteration needed for linear partial differential equations.
2. Solution is displayed as a function of one spatial dimension.
3. Conditionally stable.¹⁵
4. Solution speed is limited only by the digital computer.
5. A general executive program may be written but it must receive scaling information prior to execution. Rescaling is necessary when there is a change in Δt , Δx , Δy or the nonlinearity.
6. Digital memory requirement is confined to storing one past time plane (or one dimension less than the Component-Sharing method).

As for the hybrid implementation of the Alternating-Direction Implicit Method, it is too new to be judged. An exhaustive list of its properties can be obtained only after someone studies it carefully, in a hybrid environment. However, some of its characteristics are evident:

Hybrid alternating-direction implicit method

1. No iterations needed.
2. Unconditionally stable.
3. Solution is piecewise continuous in the time dimension.
4. High accuracy can be attained in the hybrid solution because we only integrate the change in the dependent variable using the full dynamic range of the analog computer.

Accuracy comparisons between digital and hybrid methods are not meaningful, since hybrid computer accuracy is always limited by the limited precision of its analog components. While hybrid solution times are extremely short, the programming effort is considerably greater and requires knowledge of not only analog and digital computation but of the interface problems as well.

Concluding remarks

It can be concluded that the hybrid computer solution of partial differential equations with two spatial dimensions is at best only a partial success, since it combines, to some degree, both the advantages and the disadvantages of the digital and the analog machines.

That is, the hybrid solution receives the benefit of the speed of the analog computer and the memory and the logic of the digital computer, but it suffers from the accuracy limitation of the analog and interface hardware. One firm conclusion we can draw is that the hybrid approach is justifiable only when the program is intended to be run many times, in order to offset the investment in programming and checkout. These conclusions are not startling in any way. In fact, since similar results had been obtained for hybrid computer solution of one-dimensional partial differential equations, the results we obtained were expected.

Unfortunately, it appears that hybrid computers are not a panacea for the difficulties of partial differential equations, which continue to challenge both analysis and computation.

REFERENCES

- 1 W J KARPLUS
Analog simulation-solution of field problems
McGraw-Hill New York 1958
- 2 G A BEKEY W J KARPLUS
Hybrid computation
John Wiley New York 1968
- 3 S- K CHAN
The serial solution of the diffusion equation using non-standard hybrid techniques
IEEE Trans on Computers Vol C-18 No 9 1969
- 4 H WITSENHAUSEN
Hybrid solution of initial value problems for partial differential equations
MIT Electronic Systems Lab Report No 8 1964
- 5 R VICHNEVETSKY
A new stable computing method for the serial hybrid computer integration of partial differential equations
Proc SJCC 1968
- 6 H H HARA W J KARPLUS
Application of functional optimization techniques for the serial hybrid computer solution of partial differential equations
Proc FJCC 1968
- 7 W D LITTLE
Hybrid computer solutions of partial differential equations by Monte Carlo method
Proc FJCC 1968
- 8 H HANDLER
High-speed Monte Carlo technique for hybrid computer solution of partial differential equations
PhD Dissertation Electrical Engineering Department University of Arizona 1967
- 9 R M HOWE S K HSU
Preliminary investigation of a hybrid method for solving partial differential equations
Applied Dynamics Report 1967
- 10 J DOUGLAS JR
On the numerical integration of $(\partial^2 u / \partial x^2) + (\partial^2 u / \partial y^2) = \partial u / \partial t$ by implicit methods
J Soc Indust Appl Math Vol 3 No 1 1955
- 11 D W PEACEMAN H H RACHFORD JR
The numerical solution of parabolic and elliptic differential equations
J Soc Indust Appl Math Vol 3 No 1 1955
- 12 K A BISHOP
Hybrid computer implementation of the alternating direction implicit procedure for the solution of two-dimensional parabolic partial differential equations
AIChE Journal Vol 16 No 1 1970
- 13 L N CARLING
Hybrid computer solution of heat exchanger partial differential equations
Annale de l'Association Internationale pour le Calcul Analogique (AICA) 1968
- 14 R VICHNEVETSKY
Serial solution of parabolic partial differential equations; the decomposition method for nonlinear and space-dependent problems
Simulation 1969
- 15 M T UNG
Hybrid solutions to parabolic partial differential equations with two spatial dimensions
Ph D Dissertation Electrical Engineering Department University of Southern California 1970
- 16 J DOUGLAS JR C M PEARCY
On convergence of alternating direction procedures in the presence of singular operators
Numerische Mathematik Vol 5 1963
- 17 C PEARCY
On convergence of alternating direction procedures
Numerische Mathematik Vol 4 1962
- 18 J DOUGLAS JR
Alternating direction methods for three space variables
Numerische Mathematik Vol 4 1962
- 19 J DOUGLAS JR A O GARDER C PEARCY
Multistage alternating direction methods
SIAM J Numer Anal Vol 3 No 4 1966

Numerical solution of partial differential equations by associative processing

by P. A. GILMORE

Goodyear Aerospace Corporation
Akron, Ohio

INTRODUCTION

In a number of recent articles the application of associative processing to a variety of data processing problems has been considered.^{1,2,3,4,5,6} In this paper we consider the application of the parallel arithmetic capability offered by associative array processors to the numerical solution of partial differential equations. A set of equations concerned with weather forecasting is selected as a representative problem to show the methodology used in applying associative processing technology. An associative array processor implementation of a numerical solution to the equations by a time-marching process is developed for a 50×50 mesh and corresponding execution times are given. The solution yields the time-dependent behavior of three time and space-dependent variables which represent x and y components of wind velocity and height of a constant pressure surface. The associative array processor employed is the Goodyear Aerospace STARAN IV. Since its organization and operation are not widely known, the next section of this paper is devoted to a description of the associative array processor and its operation.

THE ASSOCIATIVE ARRAY PROCESSOR

General characteristics

The Goodyear-developed Associative Array Processor (AP) is a stored program digital computing system capable of operating on many data items simultaneously; both logical and arithmetic operations are available. The principal components of an AP system are as follows: (1) An *associative memory* array (AM) in which are stored data on which the AP operates. Typically, the AM may consist of 4096 words, each of 256 bits. (2) A *response store* configuration for each

word of the AM, which provides arithmetic capability, read/write capability, and indication of logical operation results. (3) An (optional) *funnel memory* of as many words as the AM, each word of 32 to 128 bits, depending on user need. The funnel memory provides the AP system with both high speed temporary storage and high speed I/O to external devices. Data transfer between the AM and funnel memory is on a serial-by-bit, parallel-by-word basis; data transfer between the funnel memory and external devices is on a parallel-by-bit, serial by word basis. (4) A *data/instruction* memory in which are stored the AP program, i.e., the list of instructions executed by the AP, and data items required by (or generated by) the AP but not maintained in the AM. (5) A *control* unit which directs the AP to execute the instructions specified by the AP program; this control unit is similar to control units found in conventional computers. Communication channels are provided between the data/instruction memory and the sequential control unit and between both of these units and external devices.

One other unit of the AP must be mentioned here, that unit is the *comparand register* (CR). The CR may contain as many bits as an AM word and is used both to transmit data into the AM in a parallel-by-bit, serial-by-word basis and to specify masking conventions for AP operations.

A simplified representation of the AP is given in Figure 1.

AP operations

We are principally concerned with the parallel execution of arithmetic operations in the AP and, to a lesser extent, with internal transfer of data. An understanding of the AP's parallel arithmetic capability can be facilitated by considering Figure 2 which depicts the word/field structure of a hypothetical 10 word, 20 bit

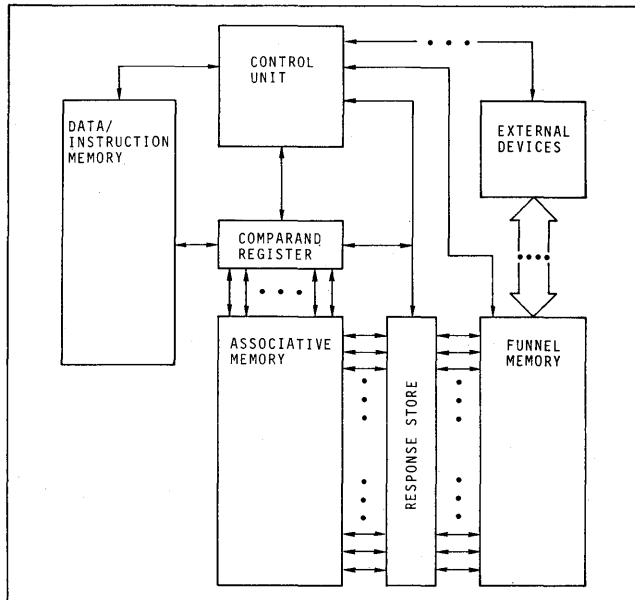


Figure 1—Simplified AP structure

AM. (For a full discussion of the AP's logical and arithmetic capability the reader is referred to Reference 7.)

In Figure 2, each of the 20 bit words has been (arbitrarily) divided into two 5 bit fields and one 10 bit field. Other field assignments could have been made and they need not be the same for all words. Field specifications are made by the programmer in accordance with computational and storage requirements at a given stage of a program; the specification is logical, not physical and can be changed for any or all words at any point in the program by the programmer.

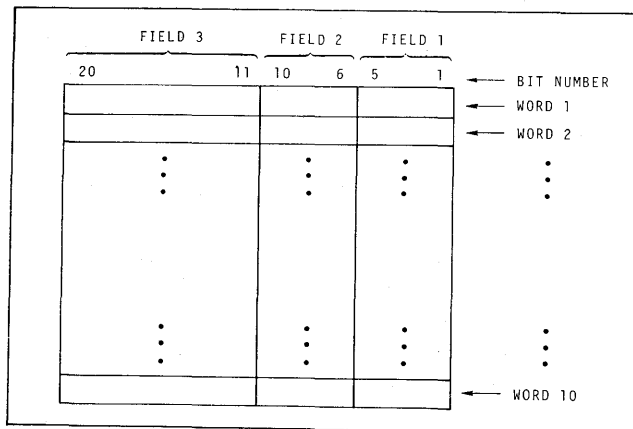


Figure 2—AM structure

If, for $i=1, 2, \dots, 10$, we denote word i by w_i ; the contents of field 1 of w_i by $(F1_i)$; the contents of field 2 by $(F2_i)$; and the contents of field 3 by $(F3_i)$, then (at least) the following computations can be done in "parallel," that is, simultaneously by word, sequential by bit. (Such parallel computations are often called "vector" computations, since they involve like operations on corresponding elements of two vectors of operands.)

$$(F1_i) \oplus (F2_i) \quad \text{or} \quad (F2_i) \oplus (F1_i)$$

$$i=1, 2, \dots, 10 \wedge \oplus \epsilon \{+, -, *, \div\}$$

The field into which the results of the operations are stored is specified by the programmer. For example, the results of the \pm operations could be stored in either Field 1, Field 2, or Field 3. We denote this, for example, by:

$$(F1_i) \pm (F2_i) \rightarrow F1_i \quad i=1, 2, \dots, 10$$

or

$$(F1_i) \pm (F2_i) \rightarrow F2_i \quad i=1, 2, \dots, 10$$

or

$$(F1_i) \pm (F2_i) \rightarrow F3_i \quad i=1, 2, \dots, 10$$

In the first two specifications, the original values $(F1_i)$ or $(F2_i)$, respectively, would be destroyed; in the third specification $(F1_i)$ and $(F2_i)$ would be unaltered.

In $*$ or \div operations, a double length product or quotient will be available. To save the double length result we would be restricted to placing the result in the "double length" field, F3. For example,

$$(F1_i) * (F2_i) \rightarrow F3_i \quad i=1, 2, \dots, 10$$

The original values $(F1_i)$, $(F2_i)$ would be unaltered.

Operations such as those described above are referred to as "within word" arithmetic operations. We have also available "register to word" operations and "between word" operations.

In register to word operations, the contents of a specified field of the comparand register, denoted by (CR) , is used as an operand. A typical register to word operation would be:

$$(CR) * (F1_i) \rightarrow F3_i \quad i=1, 2, \dots, 10$$

or

$$(CR) \pm (F2_i) \rightarrow F2_i \quad i=1, 2, \dots, 10$$

In between word operations, the operand pairs derive from different words. For example, in the operation

$$(F1_i) * (F1_{i+2}) \rightarrow F3_i \quad i=1, 2, \dots, 8$$

field 1 of word 1 is multiplied by field 1 of word 3 and the result placed in field 3 of word 1; field 1 of word 2 is multiplied by field 1 of word 4 and the result placed in field 3 of word 2 . . . ; field 1 of word 8 is multiplied by field 1 of word 10 and the result placed in field 3 of word 8. We could likewise specify an operation such as

$$(F1_i) + (F2_{i+1}) \rightarrow F1_i \quad i = 1, 2, \dots, 9$$

We note that for between word operations, the "distance" between words from which operand pairs are derived is constant, that is, with each word i , we associate a word $i \pm \Delta$.

Such between word operations are executed in parallel but are more time consuming than within word or register to word operations. The increase in time is proportional to the distance Δ .

In the preceding examples, operand pairs were derived from either AM word/field locations or the comparand register and results were stored in AM word/field locations. For AP systems incorporating a funnel memory, one element of each operand pair can be derived from the funnel memory and results can be stored in the funnel memory; operations taking place, as before, in parallel. Simple data transfer operations between the AM and the funnel memory of course proceed in a word parallel, bit serial fashion.

As the reader may suspect, the bit serial nature of AP operations results in long execution times if computation is considered on a per word basis. The source of computational advantages for an AP lies in the AP's ability to do many, indeed thousands, of operations in a word parallel fashion and thus give, for properly structured computations, effective per word execution times which are very attractive as we shall see.

In the following section we shall show how computations required in the weather forecasting problem can be structured to take advantage of the AP's parallel arithmetic capability.

A SIMPLIFIED WEATHER FORECASTING PROBLEM

Problem statement

The National Oceanic and Atmospheric Agency (NOAA), a Federal Agency which includes among its responsibilities the development of methods of weather forecasting, has provided Goodyear Aerospace with a math model⁸ which is a simplified version of the math model currently used in weather forecasting computations. The math model consists of a system of difference equations which are the discrete version of a system of

partial differential equations. The simplified model is both interesting and useful in that it does in fact provide a representative propagation problem and the computations required for solving the problem are typical of the computations required for the model actually used in weather forecasting. We shall show that such computations may be structured for effective AP execution.

The equations in the math model involve three time and space dependent variables $u = u(x, y, t)$; $v = v(x, y, t)$; $h = h(x, y, t)$ which give, respectively, the x component of wind velocity; the y component of wind velocity; and the height (above some reference) of a surface of constant barometric pressure.

The system of differential equations is given by:

$$\begin{aligned} \frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} + v \frac{\partial u}{\partial y} - fv + g \frac{\partial h}{\partial x} &= 0 \\ \frac{\partial v}{\partial t} + u \frac{\partial v}{\partial x} + v \frac{\partial v}{\partial y} + fu + g \frac{\partial h}{\partial y} &= 0 \\ \frac{\partial h}{\partial t} + u \frac{\partial h}{\partial x} + v \frac{\partial h}{\partial y} + h \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} \right) &= 0 \end{aligned} \quad (1)$$

The corresponding difference equations are:

$$\begin{aligned} u_t &= - \left\{ \overline{\bar{u}^{xy} \bar{u}_x^y + \bar{v}^{xy} \bar{u}_y^x + f \bar{v}^{xy} + g \bar{h}_x^y} \right\}^{x_y} \\ v_t &= - \left\{ \overline{\bar{u}^{xy} \bar{v}_x^y + \bar{v}^{xy} \bar{v}_y^x - f \bar{u}^{xy} + g \bar{h}_y^x} \right\}^{x_y} \\ h_t &= - \left\{ \overline{\bar{u}^{xy} \bar{h}_x^y + \bar{v}^{xy} \bar{h}_y^x + \bar{h}^{xy} (\bar{u}_x^y + \bar{v}_y^x)} \right\}^{x_y} \end{aligned} \quad (2)$$

The subscripts indicate partial derivatives; superscripts denote spatial averaging; f and g are constants. These notational conventions are those of Dr. Shuman of NOAA.

The equations are to be solved over an $n \times n$ (say 50×50) uniform square mesh with spacing "d." Initial conditions specify, at each point of the mesh, values of u , v , and h at time t_0 . The solution is specified by a marching (in time) procedure in which we approximate, for each variable, its time derivative at time $t_k = t_0 + k\Delta t$ and then predict its value at time t_{k+1} by a truncated Taylor series. For example:

$$u(t_{k+1}) = u(t_k) + \Delta t u_t(t_k) \quad (3)$$

An alternate marching method proceeds by computing, based on initial values at time t_0 , values at time $t_0 + \Delta t = t_1$, and then proceeding in a "leap frog" technique. For example:

$$u(t_{k+1}) = u(t_{k-1}) + 2\Delta t u_t(t_k) \quad (4)$$

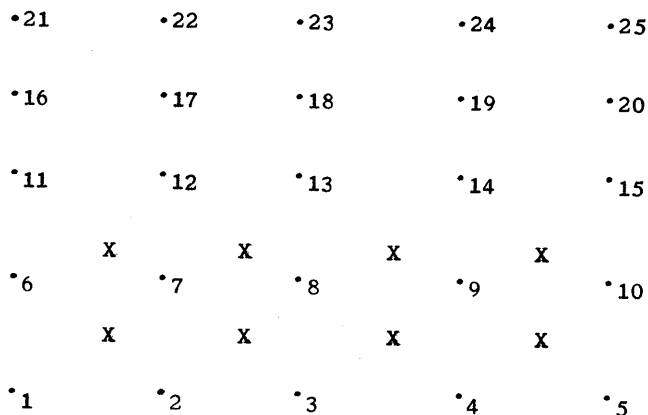


Figure 3—5x5 mesh

Computations involved in the two marching methods are nearly identical; storage requirements for computer implementations differ in that in the first method, given by (3), a set of values for the variables u, v and h at only one time period is saved from step to step while the second method given by (4), requires saving sets of values at two time periods at each step. Selection of the second method may be dictated by numerical accuracy and stability considerations.

Whatever marching procedure is employed, it is executed for each variable u, v and h at each interior point of the mesh, throughout the forecast period.

In order better to expose the computational requirements of a marching procedure we shall in the following item consider an illustrative example based on a 5x5 mesh.

Computation example

For purposes of example we shall consider a solution of the simplified weather prediction problem over a 5x5 mesh having the mesh point ordering convention of Figure 3.

We shall employ marching method (3), the resulting computations, programming, and timing being but little changed if method (4) is selected.

For each variable u, v and h , the time derivative computation at each interior point of the mesh of Figure 3 will proceed in the following fashion. Consider point 7 which may be viewed either as lying at the center of the 4 surrounding points {2, 12, 6, 8} or viewed as lying at the center of the 4 surrounding "boxes" specified by the point sets {1, 2, 6, 7}, {2, 3, 7, 8}, {6, 7, 11, 12} and {7, 8, 12, 13}. The centers

of these boxes are denoted by the x 's of Figure 3. We shall choose the latter point of view and at each of the 4 box centers (and for each variable) compute an approximation to the time derivative. The 4 computed values will then be averaged and taken as the time derivative value at point 7. This procedure is repeated for each interior point of the mesh.

Within each box the time derivative will be computed in terms of certain space derivatives, as indicated by Equation (2). For example, in computing u_t , the approximate time derivative of u , the approximate space derivative u_x is required. The calculation of space derivatives is specified in a box-to-box fashion as follows. Consider the box {1, 2, 6, 7}. For this box center, the derivative of u with respect to x can be approximated by either

$$u_x \cong \frac{u_2 - u_1}{d},$$

or

$$u_x \cong \frac{u_7 - u_6}{d}$$

where u_i denotes the value of u at mesh point i , and d is the mesh spacing. If we average the two approximations in the y direction, we have a better approximation, namely

$$\bar{u}_x^y = \frac{1}{2}((u_2 - u_1)/d + (u_7 - u_6)/d)$$

Similarly, for the rest of the boxes formed by the first two rows of mesh points we have, moving left to right, the following approximations which we refer to as "Form 1."

$$\bar{u}_x^y = \frac{1}{2}((u_3 - u_2)/d + (u_8 - u_7)/d)$$

$$\bar{u}_x^y = \frac{1}{2}((u_4 - u_3)/d + (u_9 - u_8)/d)$$

$$\bar{u}_x^y = \frac{1}{2}((u_5 - u_4)/d + (u_{10} - u_9)/d) \quad \text{"FORM 1"}$$

Other space derivatives can be approximated in like fashion.

It is evident that the computations of "Form 1" are amenable to parallel execution. Each of the differences $(u_2 - u_1), (u_7 - u_6), (u_3 - u_2), \dots$ can be computed independently and hence in parallel; subsequently, the quotients and sums may be computed in parallel. Other space derivatives and averages appearing in Equation (2) similarly involve potentially parallel arithmetic operations. The question then arises as how best to exploit the parallelism inherent in the computations by proper implementation of the computations on a computer system such as the AP which

offers parallel arithmetic capability. In the following section an AP implementation of Equation (2) for parallel execution will be developed.

ASSOCIATIVE PROCESSOR IMPLEMENTATION

In the preceding section it was seen that the computations involved in solving Equation (2) by a marching process offer the potential for parallel execution. In this section we consider the question as how best to store the required data in an AP and arrange the computations. We again use for example purposes a marching process for Equation (2) over the mesh of Figure 3.

One could associate with each point of the mesh one AP word, and in designated fields of that word store current point values of the variables u , v , and h and results of intermediate computations required to compute approximate time derivatives u_t , v_t , and h_t used in the marching process. Such a scheme would, however, have certain disadvantages as can be seen by examining the "Form 1" computations previously specified for the \bar{u}_x^y space derivative computations. With such a storage scheme the differences $(u_2 - u_1)$, $(u_7 - u_6)$, $(u_3 - u_2)$, ... can indeed be executed in parallel, but at the expense of between-word arithmetic operations in the AP; the subsequent division operation will require a register-to-word arithmetic operation followed (or preceded) by another between-word operation for the add (between words 5-distant in the 5×5 mesh example; between words n -distant in a general $n \times n$ mesh problem). Between-word operations (especially between-distant words) should be minimized for optimal performance and further analysis of operations required for the marching process if the one mesh point per word storage scheme is employed reveals that excessively many between-word operations are required. Another storage scheme is suggested by the following considerations.

The $\{\bar{u}_x^y\}$ calculations specified previously for the four "boxes" formed by the first two rows of the mesh can be rewritten, respectively, in what we shall call "Form 2" as follows:

$$\begin{aligned} \bar{u}_x^y &= ((u_2 + u_7)/2d - (u_1 + u_6)/2d) \\ \bar{u}_x^y &= ((u_3 + u_8)/2d - (u_2 + u_7)/2d) \\ \bar{u}_x^y &= ((u_4 + u_9)/2d - (u_3 + u_8)/2d) \\ \bar{u}_x^y &= ((u_5 + u_{10})/2d - (u_4 + u_9)/2d) \quad \text{"FORM 2"} \end{aligned}$$

If we were to define two vectors by $U_1 = (u_1, u_2, u_3, u_4, u_5)$

AP WORD FIELD						AP WORD	
...	6	5	4	3	2		
	h_6	h_1	v_6	v_1	u_6	u_1	1
	h_7	h_2	v_7	v_2	u_7	u_2	2
	h_8	h_3	v_8	v_3	u_8	u_3	3
	h_9	h_4	v_9	v_4	u_9	u_4	4
	h_{10}	h_5	v_{10}	v_5	u_{10}	u_5	5
					u_{11}	u_6	6
	u_{12}	u_7	7
	u_{13}	u_8	8
	u_{14}	u_9	9
					u_{15}	u_{10}	10
					u_{16}	u_{11}	11
	u_{17}	u_{12}	12
	u_{18}	u_{13}	13
	u_{19}	u_{14}	14
					u_{20}	u_{15}	15
	h_{21}	h_{16}	v_{21}	v_{16}	u_{21}	u_{16}	16
	h_{22}	h_{17}	v_{22}	v_{17}	u_{22}	u_{17}	17
	h_{23}	h_{18}	v_{23}	v_{18}	u_{23}	u_{18}	18
	h_{24}	h_{19}	v_{24}	v_{19}	u_{24}	u_{19}	19
	h_{25}	h_{20}	v_{25}	v_{20}	u_{25}	u_{20}	20

Figure 4—Redundant AP storage scheme

and $U_2 = (u_6, u_7, u_8, u_9, u_{10})$ then it is readily seen that the add operations of Form 2 are just those of the vector sum $U = U_1 + U_2$. The subsequent divisions are given by $U \star = \frac{1}{2}dU$ and, finally, the space derivatives $\{\bar{u}_x^y\}$ are given by a "convolution" of the vector $U \star$ in which we subtract from the i th element of $U \star$ the $(i-1)$ th element, $i=2, 3, 4, 5$. Such vector operations are well suited to AP execution and suggest a storage scheme in which the data items $\{u_i, v_i, h_i\}$, $i=1, 2, \dots, 25$ (for the example) are stored in a redundant fashion as follows. In two fields of words 1 through 5 we store respectively not only u_1 through u_5 but also u_6 through u_{10} . Then the two fields of words 1 through 5 will contain respectively the vectors U_1 and U_2 previously defined. In like fashion we shall store in 4 more fields of words 1 through 5 values for v_1 through v_{10} and h_1 through h_{10} . The first five words of the AP then contain all the values needed to compute time derivatives for the variables u , v , and h in the first row of "boxes" determined by the first two rows of the mesh. We can continue in this fashion by storing in second 5 words of the AP, words 6 through 10, values of u , v , and h required for computation of time derivatives in

the second row of boxes determined by the second and third rows of the mesh. We continue in like fashion till all mesh point values are stored. For the example problem this storage scheme is shown explicitly in Figure 4. In this scheme data are stored redundantly; the increased storage requirements being tolerated in return for increased computational speed. The increase in computational speed is gained by minimizing between-word operations in the AP and maximizing parallel computations. The parallel nature of the computations allowed by the redundant storage scheme is readily seen by considering Figure 5 in which is exhibited the three step sequence for computing $\{-\bar{u}_x^y\}$ for the first row of boxes. ($-\bar{u}_x^y$ is computed since that is the quantity actually required in the time derivative computations; see Equation (2). It will be noted that the three steps correspond to the three vector operations previously described. The addition operation of the first step is a within-word AP operation which is done in parallel for all data pairs (u_1, u_6) , (u_2, u_7) , ..., (u_5, u_{10}) ; the division operation of the second step is a register-to-word AP operation which is done in parallel for the intermediate data items (u_1+u_6) , (u_2+u_7) , ..., (u_5+u_{10}) . The third and final step of the \bar{u}_x^y calculation is a between-word AP operation (words a distance of only 1 away) which is done in parallel for the intermediate data items $(u_1+u_6)/2d$, ..., $(u_5+u_{10})/2d$. That the $\{\bar{u}_x^y\}$ computations for the first row of boxes can be done in parallel is evident. But a moment's reflection reveals that $\{\bar{u}_x^y\}$ computations for all boxes can be done in parallel because of the redundant storage scheme. The \bar{u}_x^y computations for the second row of boxes are of the same form as those for the first row, the only difference being that they are based on values u_6, u_7, \dots, u_{15} which comprise the values of the variable u over the second and third rows of the mesh.

Due to the redundant storage scheme, these values are stored in words 6 through 10. Likewise, in words 11 through 15, and 16 through 20 are stored, respectively, values for the third and fourth, and fourth and fifth rows of mesh points. The $\{\bar{u}_x^y\}$ computations for the corresponding boxes are again of the same form as for the first row of boxes. We have then that under the redundant storage scheme the \bar{u}_x^y computation for all boxes is specified by the same fieldwise operations and that within each of the three computational steps the computations for all boxes can be carried on in parallel. We note here that this is true not just for our 5×5 example, but for arbitrary $n \times n$ meshes, and AP effectiveness increases with increasing mesh size.

The same analysis applies to computations of spatial derivatives for the other variables v and h whose values

over the mesh are likewise redundantly stored. The sets $\{\bar{v}_x^y\}$ and $\{\bar{h}_x^y\}$ can, in turn, be calculated in parallel. Like parallelism exists for computation of space derivatives with respect to y , averaged in the x direction, say $\{\bar{u}_y^x\}$ and spatial averages, say $\{\bar{u}^y\}$. We see this by noting that, for the first box,

$$u_y \cong (u_6 - u_1)/d;$$

$$u_y \cong (u_7 - u_2)/d$$

$$\bar{u}_y^x = ((u_6 - u_1)/2d + (u_7 - u_2)/2d)$$

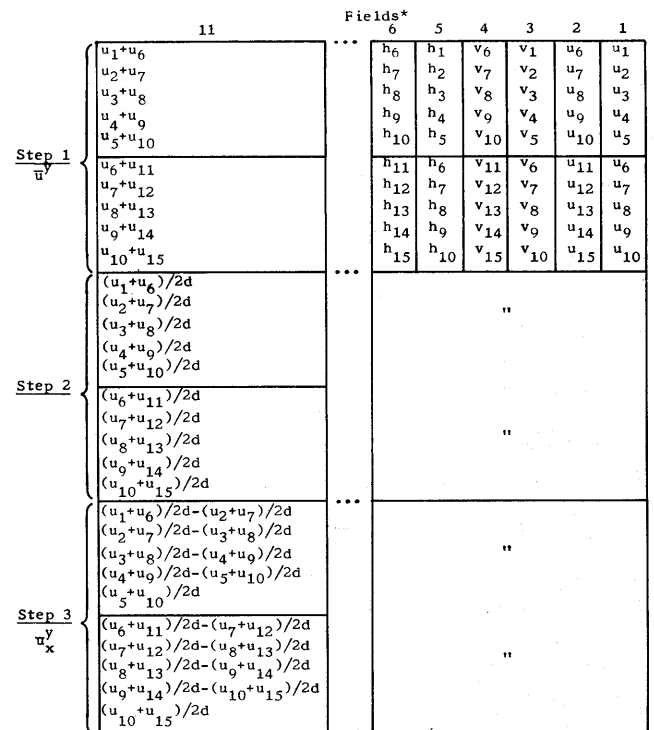
similarly, for the remainder of the first row of boxes

$$\bar{u}_y^x = ((u_7 - u_2)/2d + (u_8 - u_3)/2d)$$

$$\bar{u}_y^x = ((u_8 - u_3)/2d + (u_9 - u_4)/2d)$$

$$\bar{u}_y^x = ((u_9 - u_4)/2d + (u_{10} - u_5)/2d)$$

The respective spatial averages for the first row of



* Complete field designation is as follows:

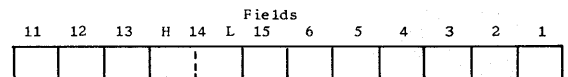


Figure 5— \bar{u}_x^y computation

boxes would be simply

$$\bar{u}^{xy} = ((u_1 + u_6)/4 + (u_2 + u_7)/4)$$

$$\bar{u}^{xy} = ((u_2 + u_7)/4 + (u_3 + u_8)/4)$$

$$\bar{u}^{xy} = ((u_3 + u_8)/4 + (u_4 + u_9)/4)$$

$$\bar{u}^{xy} = ((u_4 + u_9)/4 + (u_5 + u_{10})/4)$$

For these computations too we have parallel execution for all boxes; this is evidently true also for the variables v and h .

The actual computation of the approximate time derivatives involves not only computing spatial derivatives and averages for the variables u , v and h over the mesh, but combining these in the fashion indicated by Equation (2). The combining operations too are amenable to parallel execution. An indication of this is seen by considering the product $\bar{u}^{xy}\bar{u}_x^y$ required in computing u_t . For each box we may store computed values for \bar{u}_x^y and \bar{u}^{xy} in the AP word whose index is the same as the lowest indexed point of the box corners (i.e., with each box we associate its lower left corner). For the example computation we have in fact done this for \bar{u}_x^y and as the spatial averaging equations indicate we can do the same for the computation of the \bar{u}^{xy} , using the original storage scheme of Figure 4. For each box then we have the corresponding values for \bar{u}_x^y and \bar{u}^{xy} in two fields of the AP word associated with the box and for all boxes we may compute the product $\bar{u}^{xy}\bar{u}_x^y$ in a single within-word operation. Subsequent to the computation of a value for a variable at a new time step, a between-word data transfer operation must be executed due to the redundant storage scheme employed.

An AP program has been written for one complete updating of interior mesh points. The program assumes a funnel memory of 120 bits and is patterned after the computational procedure given in Reference 9. For a 50×50 mesh, the total time required for one updating of the interior mesh points according to the difference Equation (2) is 3.5 milliseconds. This time is based on fixed point arithmetic on 20 bit fields. (Floating point is available via software and, if employed, overall execution times will typically increase by a factor of approximately 1.4.) Due to the parallel features of the AP implementation, this execution time will increase very little (within AP/funnel memory capacity) as the mesh size increases. The independence of execution time from n , the mesh size, is due to the fact that within the updating procedure only certain between-word data transfer operations are explicitly time dependent on n . This would allow a large AP of say 24,000 words of 256 bits employing a 120 bit funnel memory to execute an update of all internal mesh points of a 150×150 mesh

in a time not more than 1 millisecond greater than the 3.5 milliseconds time required for a 50×50 mesh. It is not a necessity to increase AP size as the mesh size increases, since the mesh can be divided into blocks and the updating procedure executed in a sequential by block, parallel within a block fashion.

Other AP configurations can be employed for the updating procedure. For example, an AP with no funnel memory but with two words per mesh point would give execution times close to those given above. Such a configuration can be made transparent to the user who "sees" an AP with half the number of words and twice the number of bits per word. In such a configuration execution times increase slightly because certain operations which are apparently "within word" are in fact "between word"—words 1 distant. For the example problem the total storage requirements are not increased by such a configuration but storage capacity—some of it unused—is greater than the AP/funnel memory configuration since the AM word is 256 bits as opposed to 120 bits for a funnel memory word. The relative merits of the two configurations are of course user dependent. Also, an AP with a small, say 32 bit, funnel memory could be employed in conjunction with a conventional processor (CP) which would provide temporary storage for intermediate results and perform certain minor computations such as accumulating partial sums involved in computing time derivatives. For such a configuration, the data transfer between the AP funnel memory and the CP will markedly increase execution time but will have little effect on total storage requirements. For the 50×50 mesh problem the execution time for one update of interior mesh points would be 33.5 milliseconds for an AP/CP configuration which employed an AP with a 32 bit funnel memory tied to a CP such as the CDC 6600. One can also envision future systems in which each AP word is tied through the response store to one head of a large head-per-track disk. Such a configuration should prove quite attractive for a variety of applications.

The execution times for both the AP/CP configuration and the stand alone AP compare favorably with an estimated time⁹ of 215.5 milliseconds for the same computations (that is, one complete updating of all variables at each interior mesh point) using an IBM 360-65 with array processor (2938, model 2).

REFERENCES

- 1 E E EDDEY
The use of associative processors in radar tracking and correlation
Proceedings National Aerospace Electronics Conference
1967

- 2 W C MEILANDER
The associative processor in aircraft collision prediction
Proceedings National Aerospace Electronics Conference
1968
- 3 A COSTANZO J GARRETT
Applications of associative processors in an intercept radar system
Proceedings National Aerospace Electronics Conference
1969
- 4 L C HOBBS, et al
Parallel processor systems, technologies and applications
Spartan Books New York 1970
- 5 L C FULMER W C MEILANDER
A modular plated wire associative processor
Proceedings IEEE Computer Group Conference June 1970
- 6 J A RUDOLPH L C FULMER W C MEILANDER
The coming age of the associative processor
Electronics Magazine Feb 15 1971
- 7 STARAN IV programming manual
Goodyear Aerospace Report GER-15096 1970
- 8 National meteorological center memorandum
File No 417 March 1967
- 9 Implementation of numerical weather forecasting on a 360,
Model 65 with an array processor (2933, Model 2)
IBM Report 322-1500 1967

Consistency tests for elementary functions

by A. C. R. NEWBERY and ANNE P. LEIGH

University of Kentucky
Lexington, Kentucky

INTRODUCTION

The possibility of using consistency tests to determine the quality of an elementary function subroutine has been considered by several authors.^{1,2,3} Although none of the authors thought highly of the idea, our investigations have led us to conclude that consistency tests do have a definite provable value in some situations. The error in a subroutine has two possible sources: (a) range-reduction and (b) the reduced-range approximation. For instance, in approximating the sine of a large angle one reduces the problem to that of approximating the sine or cosine of an angle in a reduced-range—perhaps $[0, \pi/4]$. Since the range-reduction process will then involve subtracting a large integer-multiple of an inaccurately represented $\pi/2$, it is clear that the reduced-range argument will be in error by a quantity which varies linearly with the original argument. Since these range-reduction errors are unavoidable and well understood, we have concentrated our efforts on consistency tests which will help to evaluate the quality of a subroutine in the reduced-range approximation. We give three examples. In each case the variable x is supposed to be within the reduced range; the tests are still valid without this condition, but there is a diminished likelihood of our bounds being realistic when x is outside the reduced range.

SINE AND COSINE TEST

We can construct a consistency test for the sine and cosine routine for $0 \leq x < \pi/4$ based on the identity $\cos^2 x + \sin^2 x \equiv 1$. Let c, s denote $\cos x, \sin x$; let $M(c), M(s)$ denote machine values for c, s , respectively, and let c', s' be the “admitted errors” in c, s , i.e., the manufacturer is prepared to admit that $|M(c) - c|$ could be as big as c' but no bigger. If values for c', s' are not available to us we have a good cause for complaint, although our test can still yield meaningful

results using assumed values for c', s' . First we compute the residual r defined by $(M(c))^2 + (M(s))^2 - 1 = r$, and we assume that $M(c)$ and $M(s)$ are each wrong by a fraction t of the admitted error, and that the signs reinforced the error to build it up to the observed residual r . Our assumption then is $(c + tc')^2 + (s + ts')^2 - 1 = r$, or

$$t^2(c'^2 + s'^2) + 2t(cc' + ss') - r = 0. \quad (1)$$

If t_m is the root of (1) of smallest magnitude we can assert that either $M(c)$ or $M(s)$ is wrong by a fraction at least $|t_m|$ of the admitted error. This is a rigorous assertion, because throughout we have given the manufacturer the benefit of the doubt by assuming that the observed residual r was the result of small errors reinforcing rather than large errors cancelling. The only trouble is that we cannot solve the quadratic (1) because we do not have trustworthy values for c, s . To get around the difficulty we define a neighboring quadratic equation

$$z^2(c'^2 + s'^2) + 2z(c'M(c) + s'M(s)) - r = 0, \quad (2)$$

and let its root of smaller magnitude be z_m ; we attempt to relate t_m, z_m . If we differentiate (1) with respect to the parameter c we obtain

$$2t \left(\frac{dt}{dc} \right) (c'^2 + s'^2) + 2 \left(\frac{dt}{dc} \right) (cc' + ss') + 2tc' = 0,$$

hence

$$\left(\frac{dt}{dc} \right) = - \frac{tc'}{(t(c'^2 + s'^2) + cc' + ss')}. \quad (3)$$

Hence dt/dc has the sign $-t$ for the chosen argument range. From (1) we see that if $r > 0$ then $t_m > 0$. Hence from (3) $(dt/dc)_{t=t_m} < 0$, i.e., the positive root of (1) decreases if we increase the value of c . But increasing the value of c is precisely what we are doing when we replace c by $M(c)$ on moving from (1) to (2), because positive r implied $M(c) > c$. Similarly, the root will also

diminish on replacement of s by $M(s)$. We conclude that for $r > 0$ the root z_m of (2) is a close lower bound for the quantity t_m . Unfortunately the case $r < 0$ is different. In this case $t_m < 0$ and from (3) $(dt/dc)_{t=t_m} > 0$. The replacement of c by $M(c)$ is like a decrease in the parameter c , implying an algebraic decrease in the (negative) value t , hence an *increase* in the magnitude of the smaller root. This gives us a bound on the wrong side of t_m . To get a bound on the right side we must replace c by a computable approximation that is guaranteed $\geq c$. Such a quantity is $M(c) + c'$; similarly we replace s by $M(s) + s'$. Hence, when $r < 0$ we have, in place of (2)

$$z^2(c'^2 + s'^2) + 2z(c'(M(c) + c') + s'(M(s) + s')) - r = 0. \tag{2'}$$

The quantity $|z_m|$, computed by (2) for $r > 0$ and by (2') for $r < 0$, is a lower bound for $|t_m|$ and we can state that at least one of the quantities c, s is wrong by at least $|z_m c'|, |z_m s'|$ respectively. If no values are given for c', s' we can choose them at will. A particularly appropriate choice in this situation is $c' = c, s' = s$. Although these might seem unreasonably large 'errors,' this will be offset by the smallness of $|t_m|$. On substitution into (1) we shall obtain

$$t^2 + 2t - r = 0, \tag{4}$$

and hence

$$t_m = -1 + (1+r)^{1/2} \approx \frac{r}{2} - \frac{r^2}{8}, \quad |r| \ll 1. \tag{5}$$

Ordinarily one is only interested in obtaining one significant figure of t_m , and it would be a grossly incorrect subroutine that yielded r -values so large that the above two terms of the binomial series did not suffice. Having obtained t_m , we can be sure that at least one of the following inequalities holds:

$$|M(c) - c| \geq |t_m| c, \quad |M(s) - s| \geq |t_m| s. \tag{6}$$

If one prefers to have the error bound measured against the observables $M(c), M(s)$ rather than against the unknowns, c, s , one can, for $z, M(z) > 0$ and $0 < T < 1$, use the following fact based on the manipulation of inequalities:

If

$$|M(z) - z| \geq Tz,$$

then

$$|M(z) - z| \geq TM(z)/(1+T). \tag{7}$$

By substituting $|t_m|$ for T and either c or s for z , the error bounds (6) can then be written in terms of observables $M(c), M(s)$. This will generally involve a slight weakening of the inequalities.

EXPONENTIAL TEST

A consistency test for an exponential routine can be made by selecting number pairs x, h , and defining

$$\frac{M(e^x)M(e^h)}{M(e^{x+h})} - 1 = r. \tag{8}$$

If the admitted error magnitudes for e^{x+h}, e^x, e^h are a', b', c' respectively then the counterpart of (1) for this problem is

$$t^2 b' c' + t(b' e^h + c' e^x + a'(1+r)) - r e^{x+h} = 0. \tag{9}$$

In order to obtain quadratics analogous to (2), (2') we modify (9) by substituting machine values for the exponentials when $r > 0$, and when $r < 0$ we replace them by machine values plus admitted errors. The analogs of (4), (5) are

$$t^2 + (3+r)t - r = 0, \quad t_m \approx (3+r)(\rho - \rho^2), \quad \rho = r/(3+r)^2. \tag{10}$$

HYPERBOLIC FUNCTIONS

As a final example we consider a slightly more complicated identity and we simplify the problem by ignoring the admitted errors and going straight to the determination of a relative error bound. For non-negative number pairs x, h we use the identity $\sinh(x+h) + \sinh(x-h) \equiv 2 \cosh h \sinh x$. If we assume relative errors of magnitude t and with signs chosen to have a reinforcing effect on the composite error, we obtain

$$\frac{(M(\sinh(x+h)) + M(\sinh(x-h)))}{2M(\cosh h)M(\sinh x)} - 1 = r = \frac{1+t}{(1-t)^2} - 1. \tag{11}$$

This leads to

$$t^2 - \frac{t(3+2r)}{(1+r)} + \frac{r}{(1+r)} = 0 \tag{12}$$

and

$$t_m \approx \left(\frac{3+2r}{1+r} \right) (\rho + \rho^2), \quad \rho = \frac{r(1+r)}{(3+2r)^2}.$$

TESTING THE TESTS

In order to see whether our tests would indeed give useful lower bounds in a practical situation, we performed several test runs on an IBM 360/65 using

standard single-precision software⁴ for the test functions and double precision for the computation and processing of residuals. A description of the tests follows:

The consistency test for the sine and cosine subprograms was run over a range of values (0, 2⁻¹⁰, π/4) for the argument *x*. The residual *r* was evaluated from the equation

$$(M(\cos x))^2 + (M(\sin x))^2 - 1 = r$$

for *r* > 0, equation (2) was solved for the root of smaller magnitude *t_m*. For *r* < 0, equation (2') was used. The admitted absolute error bounds for the cosine and sine subprograms were 1.47 × 10⁻⁷ and 1.31 × 10⁻⁷ respectively. The maximum value for |*t_m*| was found to be .512 at *x* = .700195. Therefore, at least one of cos *x* or sin *x* is wrong by at least .752 × 10⁻⁷, and .670 × 10⁻⁷ respectively. The first bad binary digit provable by this technique occurs in the 23rd bit position.

The consistency test for the exponential subprogram was run using Equation (8) to evaluate the residual *r*. Assuming relative errors of magnitude *t*, and with signs chosen to have a reinforcing effect on the composite error, we obtain

$$\frac{M(e^x)M(e^h)}{M(e^{x+h})} - 1 = r = \frac{(1+t)^2}{1-t} - 1 \quad \text{for } r > 0$$

and

$$r = \frac{(1-t)^2}{1+t} - 1 \quad \text{for } r < 0.$$

These lead to

$$t^2 + t(r+3) - r = 0 \quad \text{for } r > 0 \quad (13)$$

and

$$t^2 - t(r+3) - r = 0 \quad \text{for } r < 0, \quad (14)$$

and the root of smaller magnitude in either case is given by

$$|t_m| \approx (3+r)(\rho - \rho^2), \quad \rho = r/(3+r)^2. \quad (15)$$

The test was run over a range of values (*h*, 2⁻¹⁰, .5) for *x* and values of 2⁻⁶, 2⁻⁵, and 2⁻⁴ for *h*. The table below shows the results

<i>h</i>	Maximum <i>t_m</i>	<i>x</i>
2 ⁻⁶	.41 × 10 ⁻⁶	.02832
2 ⁻⁵	.386 × 10 ⁻⁶	.0332
2 ⁻⁴	.369 × 10 ⁻⁶	.124

The admitted value for the relative error bound is .465 × 10⁻⁶. The first bad binary digit provable by this technique occurs in the 21st bit position.

The consistency test for the sinh and cosh subprograms was run over the same grid of values for *x* and *h*. The residual *r* was calculated using Equation (11) and the lower bound for the relative error was found from Equation (12). Care was taken that (*x* - *h*) ≥ 0 for all *h*, *x*. The table below shows the results obtained.

<i>h</i>	Maximum <i>t_m</i>	<i>x</i>
2 ⁻⁶	.159 × 10 ⁻⁶	.09180
2 ⁻⁵	.212 × 10 ⁻⁶	.0625
2 ⁻⁴	.370 × 10 ⁻⁶	.0625

The admitted relative errors are 1.2 × 10⁻⁶ and 1.31 × 10⁻⁶ for sinh and cosh respectively.

The above test was rerun using the expression .5(e^{*x*} - e^{-*x*}) with the library exponential routine to synthesize sinh *x*; hence low accuracy can be expected for small *x*-values owing to the subtraction of nearly equal quantities. The following table shows the results obtained:

<i>h</i>	Maximum <i>t_m</i>	<i>x</i>
2 ⁻⁶	4.42 × 10 ⁻⁶	.02246
2 ⁻⁵	2.36 × 10 ⁻⁶	.03223
2 ⁻⁴	2.10 × 10 ⁻⁶	.08105

The errors are seen to be much larger than in the previous run. We can conclude that this consistency test is thus able to detect a 'bad' sinh subprogram.

CONCLUSION

Although we had (and still have) no reason to doubt the manufacturer's word concerning the accuracy of the subroutines supplied, we note that our tests were able to prove errors of comparable magnitude to those that the manufacturer admitted. In one case the provable error was 89 percent of the admitted error. In view of this we feel it is safe to conclude that any manufacturer whose claims are at all extravagant can almost certainly be proved wrong on the basis of carefully constructed consistency tests alone. It is also worth noting that our tests were able to show up a badly coded hyperbolic function routine, since badly coded routines of this sort are said to be in circulation.

In summary we would like to state our position on consistency tests: Firstly we do *not* advocate them as a substitute for an elaborate full-scale validation process;

we merely note that there are many routines in circulation that have evidently not passed any stringent tests at all. Since few institutions possess the funds and manpower to do their own full-scale validation, there is a legitimate market for consistency tests which can quickly and cheaply give a rough indication of quality. Secondly we believe that in the literature, particularly,¹ there has been a tendency to underrate the potentialities of the consistency test. Too much importance is sometimes attached to the fact that a subroutine can be grossly in error while exactly satisfying a mathematical identity. One can guard against being misled by this situation, partly by use of redundancy (using several different identities) and partly by common sense (i.e., by avoiding identities which the subroutine programmer is likely to have used in the given range). Cody² gives additional advice on the choice of identity. Thirdly, although we believe that consistency tests deserve to be held in higher esteem than is commonly the case, we

are anxious not to over-correct the situation. A consistency test can only provide *lower* bounds for errors. Such a test cannot possibly tell "the whole truth" about a subroutine; on the other hand, it will tell "nothing but the truth," and we think we have demonstrated that just this much information, delivered promptly and cheaply, can be very helpful.

REFERENCES

- 1 C HAMMER
Statistical validation of mathematical computer routines
Proc SJCC 1967 331-333
- 2 W J CODY
Performance testing of function subroutines
Proc SJCC 1969 759-763
- 3 J F HART *et al.*
Computer approximations
Wiley New York 1968
- 4 IBM Publication form C 28-6596-4

Laboratory automation at General Electric corporate research and development

by P. R. KENNICOTT, V. P. SCAVULLO, J. S. SICKO, and E. LIFSHIN

General Electric Company
Schenectady, New York

INTRODUCTION

The purpose of a laboratory automation system is to improve the quality and quantity of the work of the scientists who use it. An organization such as General Electric Corporate Research and Development presents a large number of problems which are capable of being solved by laboratory automation. We wish to discuss a system which was developed to solve many of these problems. We will first describe the computer facilities which are a part of the system. Next, we will describe some of the data communications equipment which has been developed to interface individual experiments to these computer facilities. Finally, we will describe three applications which illustrate the use of the laboratory automation system.

The system we have developed is designed to be sufficiently flexible to accommodate a large number of different experiments. In the design, a premium was placed on hardware which is modular and easy to assemble into a system for the automation of any particular experiment. Since we have found it advisable for the scientist or engineer to take an active part in the implementation of the application software for his experiment, a premium has also been placed on system software features which aid a person who is relatively inexperienced in computer programming to quickly create and test his software. The Research Center consists of two locations approximately three miles apart in Schenectady together with a number of outlying locations in upstate New York. Consequently, data communications assume a greater importance in our system than would be the case with a laboratory automation system dealing with a single central location.

The history of laboratory automation at Corporate Research and Development goes back to the installation in 1964 of the first on-site computer, a GE 225. The first laboratory automation experiments consisted of off-line recording of data on paper tape, processing

of the data being done on the 225 in batch mode. An on-line analog-to-digital converter was soon installed which permitted direct on-line recording and processing of analog data. With the upgrading of the 225 to a GE 265 time-sharing computer in 1966, it became possible to carry on the collection and processing of data without devoting an entire computer to the task.

By 1967 the computing requirements had outgrown the 265 facilities, and it was decided to implement a special time-sharing system on a modified GE 600. This computer uses an addressing mechanism particularly convenient for a multi-user system. At that time, the success of the early laboratory automation experiments led to the decision to add a GE/PAC[®] 4020 process control computer as a real-time peripheral processor for the 600 to handle laboratory automation work. A PDP 9 handles graphics work for the complex (Figure 1). To supplement these computer facilities a modular line of hardware devices was developed which enable a particular system to be quickly implemented.

GE 600 SYSTEM

The GE 600 serves the main computational requirements for the Center.¹ It is a large scale computer with 32 million words of random access storage. It offers the Center's engineers and scientists a convenient and flexible computer facility. A single file system serves as the data base for batch, remote batch, and time-sharing. Hence, users may access the same data files from any of these modes.

The operating system is organized as a central executive handling scheduling and input/output. It supports a number of subsystems which handle the applications work. Among these subsystems are modules imitating the standard commercial offerings of General Electric and Honeywell Information Systems as well as modules supporting systems available only in the Center.

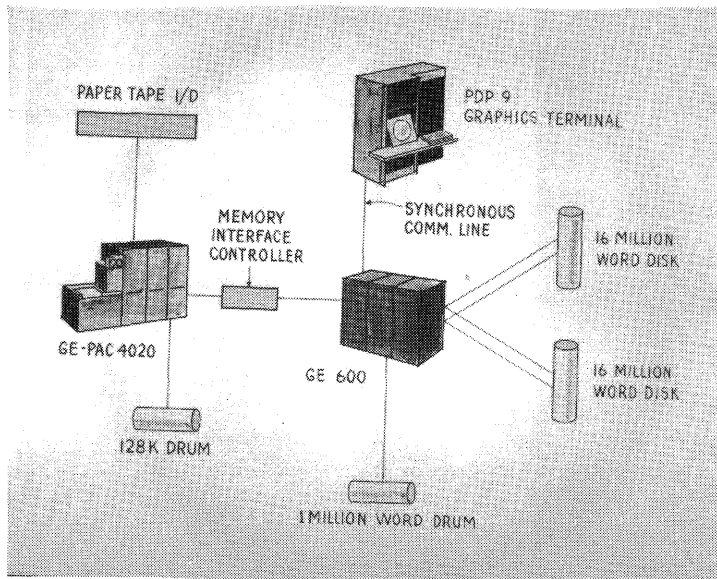


Figure 1—General Electric Corporate Research and Development computer facilities

To facilitate programming, a variety of languages have been made available. Among these are FORTRAN, BASIC, TRAC, and ALGOL. A large library of statistical, mathematical, and numerical analysis routines supplement these languages. Various editors and debugging tools are also available.

GE/PAC 4020 SYSTEM

The GE/PAC 4020 supports a data logging system which is able to log several types of data being transmitted at varied rates from many users simultaneously in real-time. In addition, it supports high-speed paper tape input/output and on-line plotting for the complete computer complex.

Data can be transmitted from an experiment to the 4020 either in digital or analog form. Communication in digital form is in the asynchronous mode and includes rates of 110, 300, 600, or 1200 baud. Provision is made for handling both forms of data either on in-house cables or over switched phone lines with the use of digital and analog modems.

The operating system in the GE/PAC 4020 is organized so as to optimize its real-time response. Thus, it will maintain its designed response time as the user load increases until it reaches the point of overload, whereas the more common type of time-sharing system simply reduces its response time gradually as the user load builds up. This design is necessary in a real-time peripheral processor such as the 4020 in order to avoid loss of data due to poor response time.

The difference in approach to response time between the two systems results in the possibility that considerable data may have to be stored in the 4020 before the 600 can store it on its disc file. The data is temporarily buffered on the 4020 drum store before transmission over the memory interface in order to accommodate this storage requirement.

Communication between the GE/PAC 4020 and the GE 600 takes place over a memory interface controller (see Figure 1). The GE/PAC 4020 can read from or write to the GE 600 core memory. Each computer can interrupt the other computer. A program exists in the 600 which handles all the requests made by the 4020. This program writes data to the disc and prepares plot and paper tape punch files.

Each computer can perform its functions independently of the other. However, since the buffering capability of the 4020 is limited, it cannot tolerate an extended outage of the 600. This is because the 4020 must eventually transmit the data it collects to the 600 for permanent storage on the disc. Once the data is stored on the disc it is available for access by user programs which run on the 600.

In order to insure real-time service for all higher speed devices on the system, an automatic double buffering technique was employed in both hardware and software. This consists of assigning two buffer control words for each device or channel which has a high speed capability. The hardware was modified to automatically switch between the two buffer control words each time an operation on one of them was satisfied. The software was organized in a way which guarantees there will always be two read operations under way for these devices at any given time. This technique increases the

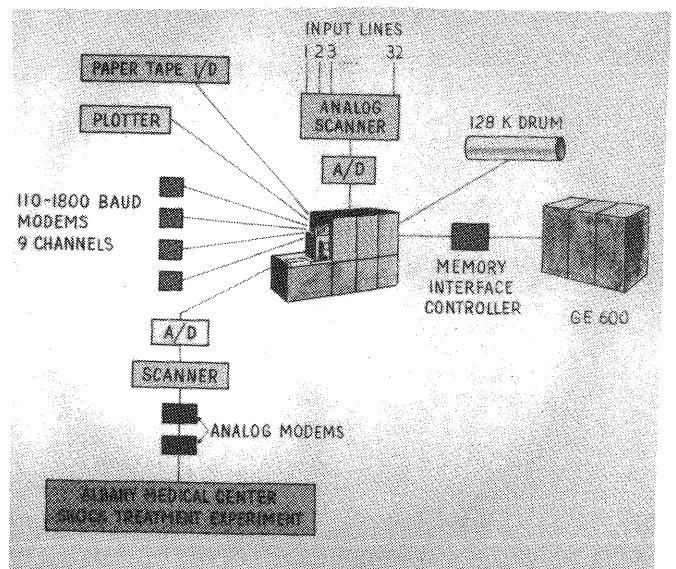


Figure 2—Data collection computer hardware

time available to replenish the buffer control word from one sample time to N times the sample time where N is the number of samples in the buffer.

GE/PAC 4020 HARDWARE

The data collection hardware is shown in Figure 2. It consists of a special low-speed analog scanner handling the analog modems described below, a set of 9 standard asynchronous communications interfaces which can be plug-adjusted to operate at speeds between 110 and 1800 baud, the on-line plotter, paper tape I/O equipment, and the high-speed analog scanner serving the experiments hardwired to the computer system.

Data in analog form is received at the GE/PAC 4020 by this high speed scanner. (See Figure 3) This peripheral was developed by the Design Engineering Group specifically for the laboratory automation system. The analog scanner was designed to place as much control as possible in hardware instead of software. Not only does this reduce software load, but, in a multi-user system, it also assures the individual user of a more uniform sampling rate for his data than would be possible in a software-driven data collection system. It is capable of accepting 120 analog lines, although only 32 have been implemented. Each line can be programmed to be turned off or to sample at one of the following data rates: 120, 60, 30, 15, 7.5, 3.75, 1.87, .937 samples per second. A specific block of core storage is dedicated to the high speed scanner control, with individual lines having their own specific locations for control words. Programming of the high speed scanner is accomplished

by storing control words in this core area for starting, stopping, or specifying the sampling rate for each line, and then initiating the transfer of the entire block to the scanner control hardware. The information so transferred controls the scanner hardware until it is updated by another transfer of the control block.

The sampling rate is under control of a master clock running at 4 MHz. The exact frequency is under servo control of the 60 Hz. line frequency. This enables the user who wishes to synchronize his experiment with the data collection system to do so by use of the power line frequency.

When the command to commence sampling a line is received by the line control logic, a status signal is raised. Actual sampling does not begin, however, until a control signal is also raised. While the status signal can be used for this control signal, it is often more convenient to carry both signals to the user's site where they furnish status indications and control capability. By the use of pulses of suitable length on the control line, it is possible for the user to cause the sampling of a single datum, continuous sampling of data, or an end-of-file indication in the hardware for his line.

The scanner has two stages of automatic gain control. The first stage, which exists in each line, has two ranges, X1 and X64. It is set by the automatic ranging logic while the previous line is being sampled, thus allowing time for settling for the somewhat less expensive line amplifiers. The second stage, which is common to all lines, has three ranges, X1, X4 and X16 giving a total of six ranges. Since the second stage must be set during the actual sampling time of a given line, it must have a faster settling time than the first stage.

The line to be sampled is connected to the common amplifier and analog-to-digital converter by a field effect transistor multiplex switch. The line must be sampled in a time sufficiently short to allow all possible lines to be sampled at their maximum rate, i.e., 120×120 samples/sec or $69.4 \mu\text{sec}$. The digital representation of the output of the common amplifier, together with the bits representing the ranging amplifier settings, is placed in core memory by cycle stealing.

The location in core where the data will be stored is controlled by a word for each line stored in the dedicated scanner control block. This word has address and tally information, and is automatically updated by the scanner hardware after each sample. When the tally for a given line runs out, an interrupt is generated for that line and hardware provides an alternate buffer. Thus, after the control information is transmitted to the scanner hardware, no additional software attention is required until the interrupt indicates that a buffer is full.

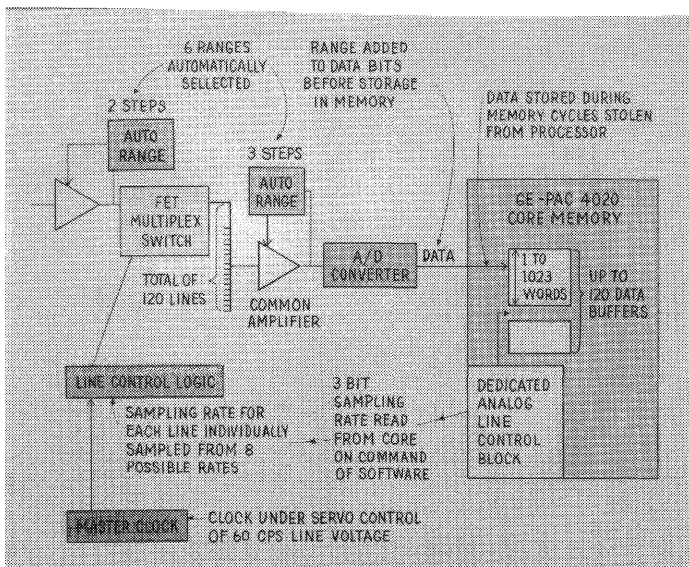


Figure 3—High speed scanner

More than one input line on the High Speed Scanner can be assigned to a single user by placing pointers to the same buffer in the control word of each line. The data from each line will be stored sequentially in the core buffer, thus allowing the sampling of several variables at once. By sampling the same variable on several lines, a sampling rate higher than the maximum 120 samples per second can be realized.

GE/PAC 4020 SOFTWARE

The operating system on GE/PAC 4020 data logging system is based on the Real-Time Multiprogramming Operating System (RTMOS) of the General Electric Process Computer Department. RTMOS is a large grouping of programs and subroutines, available only on GE/PAC computers, that supervises the interaction of process events, time, computer peripherals, and the central processor of the computer.² The core resident routines of RTMOS which perform scheduling, core management and related routines are the only parts of RTMOS used on the GE/PAC 4020 Data Logging System. An I/O system which supports time-sharing and a set of functional programs which supports the teletype command system were designed and implemented for this system. All data gathering, table manipulation, and utility programs were also developed for this system.

The I/O system was designed so that all devices on the system use common feeder, driver, and interrupt handling routines. This is accomplished by associating a table with each device type. Depending on the device type presently being serviced, the proper table is accessed and appropriate operations or subroutines are executed. In most cases, a new device can be implemented simply by creating a table for that device.

An operating system running under RTMOS is organized as a set of functional programs which can be scheduled and will run independently. In the data-logging system we have developed, two types of functional programs are found—applications programs and a type of program which supports the teletype command system called a manager program. With each manager there is associated a group of commands. For each command in a given manager's list, there is a corresponding applications program. When a command is given to a manager from a user's teletype, the applications program corresponding to that command is run by the manager. At the same time, the manager removes itself from core. When the applications program terminates, it is automatically removed from core and the manager which was previously running is brought back and run again. Any applications program under

a given manager can replace itself with another applications program from that manager's list.

The manager program concept enables new commands and their associated applications programs to be easily added to the system. Also, since one program can replace itself with another program, a program chain can be implemented.

There are three basic tables which are used by the 4020 operating system, the User ID table, the analog table, and the digital table. All users have their user identification, password, and billing number entered in the User ID table. Users who transmit data in the digital mode have an entry in the digital table. Similarly, users who transmit data in the analog mode through the high speed scanner have an entry in the analog table. The digital table indicates whether the user is a local user transmitting on a specific hard-wired line or a remote user transmitting over any available phone line. The analog table indicates which lines are assigned to a user and the sampling rate for each line. Automatic table manipulation is provided. Thus, new users can be brought on the system in a matter of minutes. Existing users of the analog system can change variables to be sampled or sampling rates simply by making a change in their table specification.

In addition to these real-time data logging functions, three utility functions are performed by the GE/PAC 4020. Data which have been produced off-line on paper tape can be sent to the 600 disc through the 100 frames/sec paper tape reader. Data which have been refined on the GE 600 can be punched on paper tape on a 120 frames/sec punch. On-line plotting of data is done on the GE/PAC 4020 to relieve the GE 600 of this task.

DATA COMMUNICATIONS

We have described the computer facilities of our laboratory automation system. A second part of the system is the data communication facilities. It is much more difficult to standardize this part of the system because of the varying needs of the individual experiment and the fact that these varying needs impact more closely on the data transmission system than on the computer facilities.

This variation of requirements from experiment to experiment is found in several forms. The output of some experiments is an analog signal such as the output of an amplifier, while the output of others is digital, such as the output of a scaler. The question of speed arises in two aspects in the design of the communications for a particular experiment. First, the rate at which data is conveniently produced by the experi-

ment dictates the equipment to be used. If the rate is slow enough to allow transmission over the 110 baud channels characteristic of teletypes, the system required is much simpler than if higher rates are required. Second, the response to the experimenter dictates the form of the communications systems. It may be possible to simply collect data and process it at the end of the experiment. This would result in a simpler system than if it were necessary to feed back processed data to influence the future course of the experiment. The amount of logic required at the experimental site is another aspect of data communications to be considered. In order to effect enough time-saving to justify the effort required for automation, it may be necessary to automate much of the control of the experiment, while for other experiments simply collecting the data is sufficient. A final aspect of data communications design is the nature of the experiment. If the experiment is to be performed only one or a few times, it will require a simpler interface than one which will be routinely performed many times. Small tasks which the experimenter does willingly a few times become onerous burdens when they must be repeated many times. The following is a description of the data communication hardware and software available to the staff which can be used with the computer facilities available at Corporate Research and Development.

We have described the high speed scanner, the analog peripheral of the GE/PAC 4020. In support of this scanner, a cabling system has been installed which runs the length of the main building at the Center. This provides cable facilities close to the experiments and makes for a convenient way for the staff to connect their experiments to the laboratory automation system. Figure 4 shows the analog transmission system.

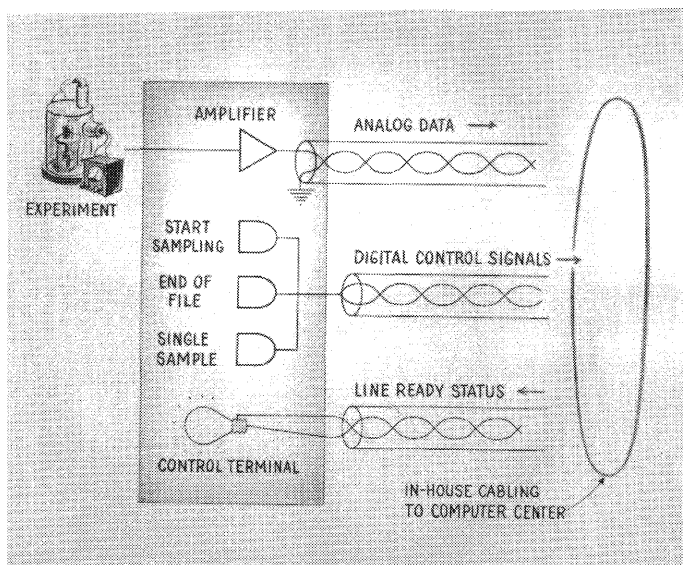


Figure 4—Analog data transmission system

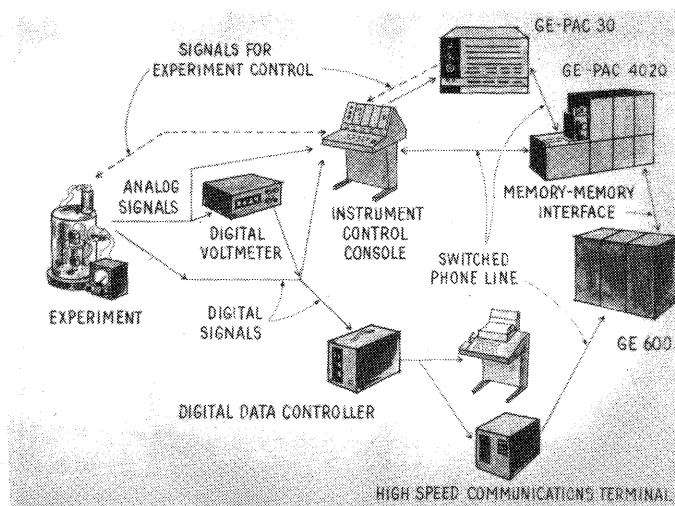


Figure 5—Digital data transmission system

Transmission is by individually shielded twisted pairs. While the user can connect to the system in a number of ways, the control terminal shown in the figure has proven convenient. Within the terminal is a light indicating the status of the line; logic to produce the appropriate pulse length for single samples, continuous samples, or end-of-file; and an amplifier with up to 60 db gain to aid in interfacing with the experiment.

Figure 5 illustrates a family of the digital modules available for data collection and control. The experiment can supply data in either analog or digital form. Analog information is converted to digital form by either a digital voltmeter or by an A/D converter in the instrumentation control console. The digital data is sent on either the instrumentation control console or to a digital data controller. The latter device formats the data for either off-line recording or transmission to a computer by the high-speed communications terminal. Off-line recording can be done with paper tape or incremental magnetic tape. The high-speed communications terminal serializes the data for transmission over asynchronous lines, and operates at either 110 or 1200 baud.

The functions of the digital voltmeter, digital data controller, and high speed communications terminal are all combined in the instrumentation control console. This device finds its greatest use when information must flow both to and from the experiment. It can be interfaced to the GE/PAC 4020 or, if local control information is to be generated, to a mini-computer. In cases where response time is not a factor, either the instrumentation control console or the high speed communications terminal can be interfaced directly to the GE 600, thus avoiding the additional step of the 4020.

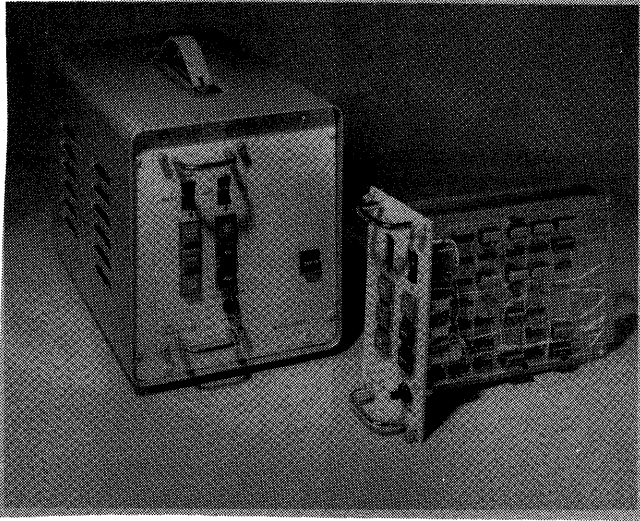


Figure 6—Digital data controller

The digital data controller interfaces a variety of digital signals to the data communication media. Figure 6 illustrates the data controller plug-in module with its housing. It can be programmed either manually or automatically. The input to the data controller is from one to eight sources of 4-wire BCD information, each having from one to eight decades. Output is eight-line ASCII code with even parity. Commas, carriage return, and line feed characters are inserted where necessary.

Figure 7 illustrates the high speed communication terminal plug-in module and its housing. The high

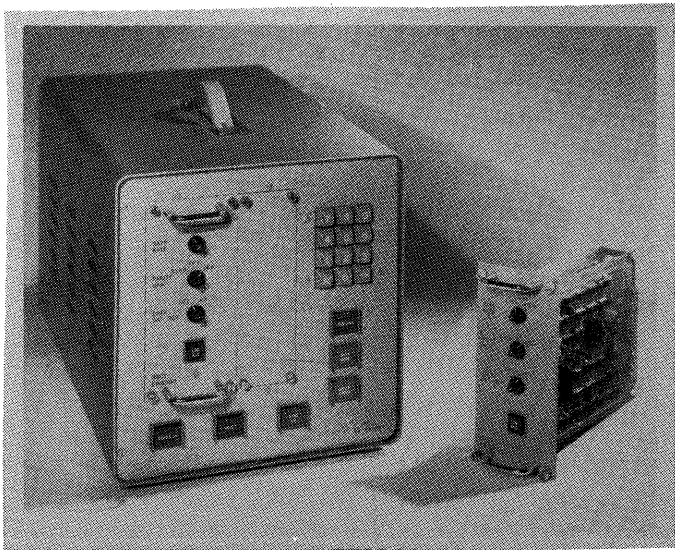


Figure 7—High speed communication terminal

speed communication terminal is used to interface the digital data controller to transmission line data sets. It formats the data into bit-serial for asynchronous transmission at either 110 or 1200 baud. It can also receive serial data from the computer and format it into bit parallel for displays, plotters, and control signals. The keyboard shown on the housing is used during the conversation mode with the 4020 Computer for 1200 baud data logging.

For those cases where it is inconvenient to use the dial-up communications network, an electronic system simulating a 1200 baud frequency shift keying system is available to be used with the in-house cabling system. The call-up for this system is accomplished by pushing a button. When the computing system responds, a light is turned on to indicate that the computer has responded to the call.

Figure 8 is an illustration of the instrumentation control console. It contains a keyboard with a 64 ASCII character set, controls for systems configuration, and displays for computer commands. The control console is designed with a pair of data busses, and will accept five separate plug-in modules. Connections to the data bus are accomplished by depressing buttons on the front panel of the console. There are two data busses in the console. The output bus accepts data from one of four sources: the digital data controller, the binary A/D converter, the keyboard, or an external device. One of three transmission/recording devices can be connected to the output bus: the high speed communication terminal, a mini-computer, or a teletypewriter. To set up a system, the operator selects one of the data sources and one of the transmission/recording media.

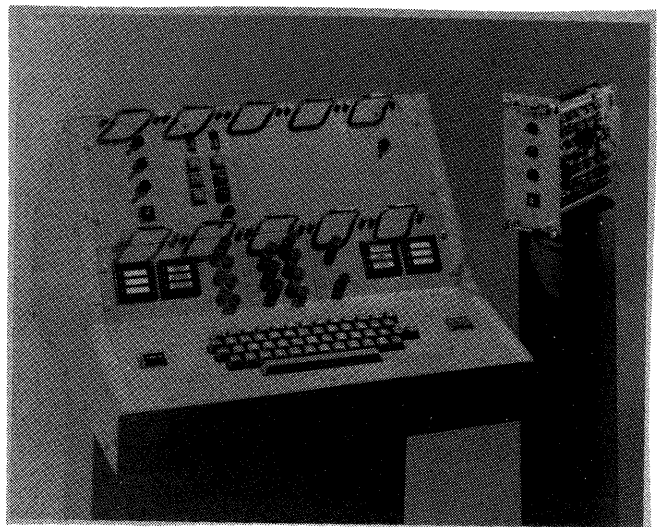


Figure 8—Instrumentation control console

This connects a data source to a transmitting system with a recording medium. In the same way, the receive bus can accept data from the high speed communication terminal, a mini-computer, a teletypewriter, or the keyboard. The receive bus can transmit data to either an external device or a digital-to-analog converter. The system controller plug-in module monitors data on the receive bus and decodes data for the console displays. The instrumentation control console with its data busses provides a convenient way to configure a data acquisition and control system. It is portable and quickly placed into operation. The plug-in module concept increases its flexibility. In addition, the ability to perform maintenance on the plug-in level increases the system up-time.

While a variety of mini-computers have been interfaced to the laboratory automation system, the majority of our experience has been with the GE/PAC 30. This is a 1 microsecond sixteen bit word machine with up to 16 k bytes of core store and an optional 65 k disc. A variety of analog or digital peripherals are available for interfacing to experiments as well as the conventional I/O peripherals. A variety of software has been written for use on the GE/PAC 30. Included are programs for data collection, graphics display, stepping motor controllers, and 110 or 1200 baud data transmission.

A peripheral which has proved useful in software development for the GE/PAC 30 is the TRICOM switch shown in Figure 9.¹⁰ This is an automatic three-way switch handling low-speed asynchronous communications lines. It allows the operator to interface the

teletype with the 600 time-sharing system in order to utilize the editing facilities for writing his program. When the program is ready the user initiates an assembly and supplies names of any library subroutines required. When assembly is complete, the resulting binary code is loaded, together with library subroutines, into a pseudo-core image of the GE/PAC 30 in 600 core. When this operation is complete, a special character switches the TRICOM to connect the 600 to the 30 for transmission of the core image. When transmission is complete, the TRICOM switch connects the teletype to the 30 for running the program. Figure 9 also shows the path of data from the GE/PAC 30 to GE/PAC 4020 to GE 600.

APPLICATIONS

The computer facilities and data communication equipment are the tangible parts of our laboratory automation system, but even more important are the people and facilities that make it work. The scientist or engineer wishing to automate his experimental work can find personnel familiar with the computer and data communication systems. These people are able to give applications advice or, if necessary, to design and implement extensions to these systems. Applications information is also provided on a variety of measurement equipment available from an instrument pool. Finally, if special equipment is required, a model shop can assist in its design and fabrication.

To automate each research or development project, a careful study is made to determine the data collection and control requirements for each experiment. Based upon the specific needs for hardware and software, in many cases a system is assembled from the available digital and analog subsystems. If the in-house terminal hardware, software, and computer systems cannot adequately meet the data collection needs, additional in-house development is undertaken or outside vendors are investigated.

When the hardware portion of a laboratory automation project is complete, the applications programs must be written. This is the responsibility of the experimenter who will benefit from the project. We have found that, in general, it is easier for a scientist to learn the necessary computer programming to implement his applications software than it is for a computer programmer to learn the necessary science to write the programs himself. This principle is particularly true when the programming is done in a time-sharing environment such as ours. Following are descriptions of three experiments which have been automated and for which the application software has been implemented

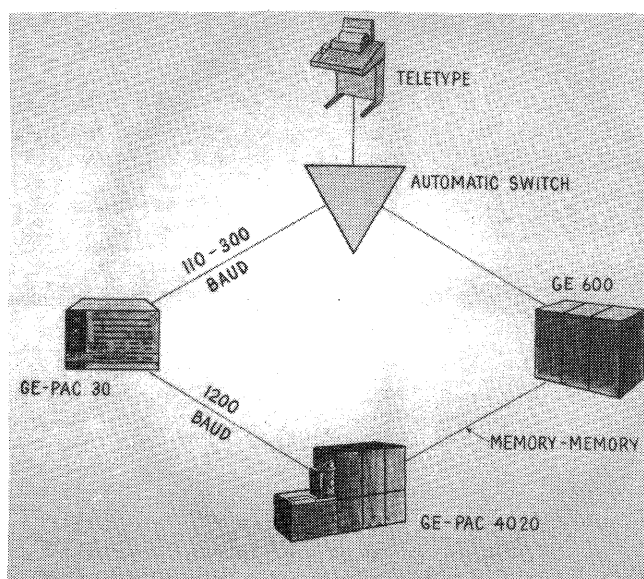


Figure 9—TRICOM switch

TABLE I—Additional Laboratory Automation

- | |
|--|
| (1) Infrared Spectroscopy |
| (2) Nuclear Magnetic Resonance |
| (3) X-ray Crystallography |
| (4) Capacitance/Voltage Measurements on Semiconductors |
| (5) Stress/Strain Relaxation Measurements |
| (6) Optical Spectroscopy |
| (7) Atomic Absorption Spectroscopy |
| (8) Electronic Micro Balance |
| (9) Gas Mass Spectroscopy |

in this manner. In addition to the three experiments defined in detail, Table 1 is a brief listing of other laboratory areas where automation has been incorporated.

SPARK SOURCE MASS SPECTROGRAPH³

The spark source mass spectrograph is an instrument for analyzing trace impurities in electrically conducting solids. In some respects it resembles the more well-known emission spectrograph, but it is approximately 1000 times more sensitive. In the course of an analysis it produces a photographic plate which contains the data from one sample (Figure 10). There are 16 exposures arranged horizontally along the plate. Each element in the sample being analyzed contributes a group of one or more lines in each exposure. The blackening of each line depends on the amount of the particular element in the sample.

It is these lines which contain the information about the sample one wishes to recover. Two aspects of the plate suggest an automated system for the recovery. First, the individual lines vary in shape from place to place on the plate. Thus, it is necessary to sample a

number of points across a given line to obtain an accurate representation of the line. Second, the variability of the photographic emulsion from plate to plate makes it necessary to derive the characteristic curve, or the relation between blackening of a line to the number of ions striking the line, from the plate itself. Fortunately, there is enough information on the plate to do this, but it entails a lengthy calculation. The automation system enables one to sample the blackening of a line on the plate in sufficient detail to obtain an accurate representation of the line, to utilize isotopic ratio information derived from the plate to obtain the derivative of the characteristic curve, to solve the resultant differential equation for the characteristic curve, and to utilize the resulting information to convert the blackenings of the various lines on the plate into concentrations of impurities in the sample. Thus, there is a requirement for a system to collect analog data. The speed with which the microdensitometer can produce data is about 10 to 15 samples per second. It is not necessary for the resulting data to be fed back to the operator during the time he is collecting it, so the relatively large calculation can be done after the collection has taken place. The application is a routine one, thus justifying some effort to make it convenient for the operator to use.

In order to use the system the operator places a plate on the carriage of the microdensitometer, moves the carriage to position the line he wishes to record on a projection screen, and presses a switch indicating to the logic that the plate is ready for scan. The computer initiates and times the scan. During the scan the operator enters on a teletypewriter the alphanumeric information necessary to describe the line being scanned. Interlocks insure that a complete set of both analog and alphanumeric information is entered for each line. Analog information is transmitted to the computer center over analog cables where it is digitized and stored on bulk storage. The teletype is connected to the computer system in a current-loop circuit. After collection of the information, it is processed and the results returned to the operator over the teletypewriter circuit.

The use of the system has resulted in an improvement in both the quantity and the quality of the work. Whereas before using the system, it required 16 hours to process the data on a plate, it now requires one hour—an improvement of a factor of 16. A group of 12 laboratories participated in a round-robin analysis in which they each analyzed the same sample of copper. The average precision of all laboratories was 40 percent, while our laboratory reported results which proved to have a precision of 15 percent, an improvement of over a factor of two.

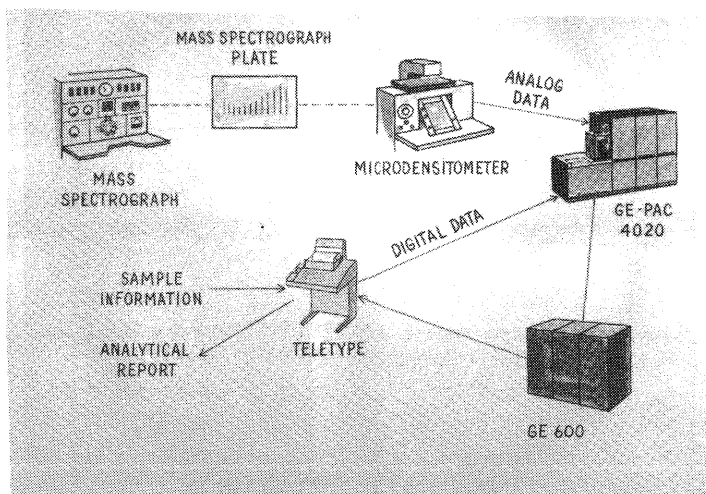


Figure 10—Spark source mass spectrograph

THE ELECTRON MICROPROBE

In the electron microprobe analyzer a beam of electrons from an electron gun is focused by magnetic lenses to a micron size spot on the surface of a specimen causing the emission of x-rays. Qualitative chemical analysis of the excited region can then be performed by scanning a crystal spectrometer to establish the wavelength distribution of the emitted radiation. Quantitative analysis involves tuning the crystal spectrometer to a specific wavelength corresponding to a particular element and counting the x-ray pulses emitted both from the specimen and a standard. Although standards of similar composition to the specimen provide the simplest method of calibration, i.e., by direct comparison, they are usually not available, particularly at the homogeneity levels required for microprobe analysis. An alternative method which needs only the use of pure elemental or simple binary standards is based on theoretical correction procedures which have been described extensively in the literature^{4,5} and relate x-ray intensity to composition by equations of the form:

$$K_A = C_A \times \left(\begin{array}{l} \text{electron} \\ \text{backscatter} \\ \text{factor} \end{array} \right) \times \left(\begin{array}{l} \text{electron} \\ \text{penetration} \\ \text{factor} \end{array} \right) \times \left(\begin{array}{l} \text{x-ray} \\ \text{absorption} \\ \text{factor} \end{array} \right) \times \left(\begin{array}{l} \text{secondary x-ray} \\ \text{fluorescent} \\ \text{factor} \end{array} \right)$$

where

K_A = the ratio of measured x-ray intensity of element A to that of an A standard with both intensities corrected for background, drift, and counter tube dead time and

C_A = the weight fraction of A .

Since all of the above factors are themselves complicated functions of composition, the calibration equation must be solved iteratively. Furthermore, intensity ratios of at least $N - 1$ components (N = the total number of components) are necessary and therefore $N - 1$ calibration equations must be solved simultaneously. Since hand calculation for even a few data points in a binary system takes hours, a computer is

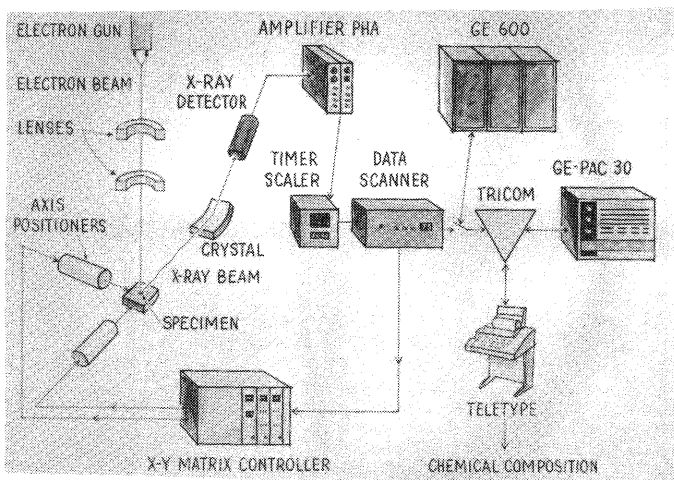


Figure 11—Electron beam microprobe system

essential for analysis of the multipoint multicomponent systems frequently encountered in practice. During the past ten years dozens of different computer programs have been written to reduce computation times to minutes even for the most complex systems.⁵

Since it is often desirable to map out the composition of a specimen an $x-y$ matrixing system was developed for our Cameca microprobe which can be used to automatically control specimen position and collect digital x-ray data as shown in Figure 11. The system shown is for a single spectrometer, but is in fact connected to four spectrometers and a digital specimen current readout. In practice the region to be analyzed is manually positioned while being observed by an optical microscope coaxial with the electron optical system. A fixed number and size of x and y steps are selected (e.g., a 3×10 matrix with x steps of 2 microns and y steps of 5 microns). Operation of the system is then initiated by a switch on the data scanner. Amplified x-ray pulses from each spectrometer are converted to pulses of fixed size and time duration which are counted by scalers for a preset time. The parallel data from the scalers is serialized by the data scanner and transmitted to the teletype where the results are printed and a paper tape punched and/or the data is transmitted to a GE/PAC 30 computer through TRICOM. The operator can then call the 600 on time-sharing and transmit the data and any other information into a data file. The 30,000 word microprobe analysis program, GEMAGIC (written by J. Colby⁷ and modified by R. Bolon⁸) can then be run in remote batch mode with the results either printed on a high speed printer in the computer room or returned on the teletype. Options are also available for plotting the results on a Calcomp plotter, an example of which has been re-

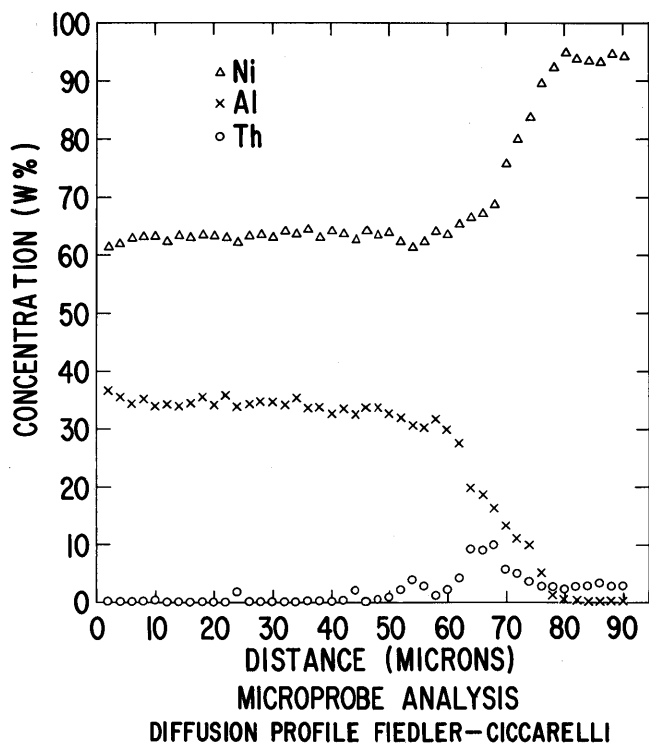


Figure 12—Electron beam microprobe diffusion scan

drawn and shown in Figure 12 for a diffusion scan across a nickel aluminum coated, thorium-doped nickel alloy.

PHYSIOLOGICAL MEASUREMENTS

The last application we wish to discuss concerns some physiological measurements being made at the Trauma Unit of Albany Medical Center Hospital.⁹ This is a single bed unit devoted to the care of, and research on, extremely seriously injured patients. Five measurements are currently being made on patients in the unit with the aid of our laboratory automation system by Dr. S. Powers and his staff. (Figure 13)

Functional residual capacity of a patient's lung is a measurement of the lung volume which is usable in respiration. The measurement is useful, for example, in reaching a decision regarding the use of a respirator for the patient. The measurement is made by abruptly switching the atmosphere being breathed by the patient from air to 100 percent oxygen and measuring the amt. of gas necessary to wash out the nitrogen from the air in the lung by oxygen. The information needed to calculate functional residual capacity is total gas flow from the lung which is measured by a positive

displacement meter and the concentration of nitrogen in the expired gas which is measured by measuring mass 28 on a small mass spectrometer sampling the expired gas.

The two signals are transmitted to the Center using analog modems over standard voice-grade telephone lines. There they are digitized by the 4020 and stored on 600 bulk storage. The calculation made on the data is an integration of the nitrogen concentration over the total volume of gas flow. To provide backup in the event of failure of the data collection system, an analog tape recording of the information is made at the same time the data are being collected.

Two other measurements are made with the same system, but with different settings of the mass spectrometer. These are oxygen uptake using mass 32 and carbon dioxide output using mass 44. These tests are typically measures of the patient's metabolism, but, in the case of shock patients, also give information regarding the patient's success in fighting trauma. As the body reacts to a loss of blood, for example, it decreases the flow of blood to the extremities, resulting in a decrease in oxygen uptake. As the body recovers, it increases circulation to the extremities, with a resulting increase in oxygen uptake.

Another measure is pulmonary venous admixture. This is the percentage of blood flowing through the lung which is shunted through non-active tissue and consequently is not oxygenated. The measurement is made by calculating a mass balance for oxygen across the lung membrane. The required data are the gaseous

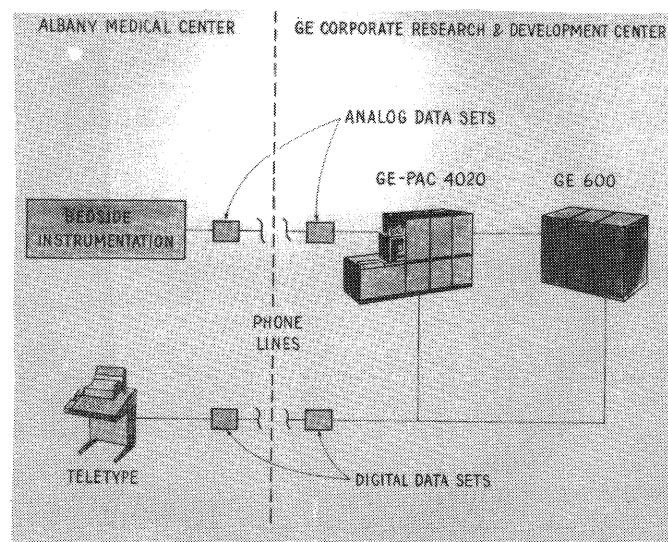


Figure 13—Physiological tests

oxygen and oxygen content of blood flowing into and out of the lung. The gaseous oxygen content is measured by the mass spectrometer system, while the blood oxygen content measurements are made with a blood oxygen analyzer.

The fifth measurement is cardiac output. This is measured using a dye-dilution technique. A dye is injected in the blood stream and the rate it is diluted is measured using a densitometer through which a portion of the patient's blood flows.

A by-product of the automation of these tests is the ability to store the results in the computer files for later retrieval. It is thereby possible to obtain a record of all tests together with the trend of their results for a given patient.

Each of these measurements is a standard physiological test, and nothing is remarkable about doing them *per se*. The interesting aspect of this work is that the tests are being made on patients actually in shock, and that the laboratory automation system aids in obtaining the results within 5 to 10 minutes from the time the tests are made. Thus the physician attending the patient can make decisions regarding the care of the patient much more quickly, and, with the aid of the additional information furnished by these tests, more accurately.

CONCLUSION

As the above applications demonstrate, the laboratory automation system has permitted experiments which would not have been possible without it. While this in itself would permit us to view it as a success, a far more important aspect of the system impresses us. The development of the system has been evolutionary and will continue to be so. There is, for example, a requirement now for synchronous communication channels between the mini-computers and the 4020. We find, however, that this evolution has been provided a framework in the form of the computer operating system and the data communications system about which to grow. Thus, a sound system design at the time these systems were first conceived has provided, and will continue to provide, direction for the orderly growth of our system into areas which we could not have foreseen at that time.

ACKNOWLEDGMENTS

We wish to acknowledge the assistance of J. Newell of Albany Medical Center Hospital in describing the physiological test application. The system we describe here would not have been possible without the contributions of many of our colleagues at the Corporate Research and Development Center. We take this opportunity of acknowledging the work of a long list of contributors.

REFERENCES

- 1 R KERR A BERNSTEIN G DETLEFSON
J JOHNSON
Overview of R+DC operating system
General Electric Report 69-C-355 1969
A J BERNSTEIN J C SHARP
A policy driven scheduler for a time sharing system
Comm ACM 14-74 1971
- 2 Anonymous
GE/PAC 4020 RTMOS manual
General Electric Company
- 3 P R KENNICOTT
A system for the quantitative evaluation of mass spectrograph plates
14th Annual Conference on Mass Spectrometry Dallas 1966
- 4 R CASTAING
Application of electron probes to local chemical and crystallographic analysis
PhD Thesis Univ of Paris France 1951
- 5 K F J HEINRICH Editor
Quantitative electron probe microanalysis
National Bureau of Standards Special Publication
Washington Vol 298 1968
- 6 D R BEAMAN J I ISASI
A critical examination of computer programs used in quantitative electron microprobe analysis
Anal Chem Vol 42 p 1540 1970
- 7 J COLBY
Quantitative microprobe analysis of thin insulating films
Adv X-Ray Anal Vol 11 p 287 1968
- 8 R BOLON
Private communication
General Electric Company
- 9 S R POWERS JR MD ET AL
Analysis of mechanism of disturbed physiology in critically ill patients
7th Annual Meeting of Soc of Engineering Science
St Louis November 3 1969
- 10 H HURWITZ
Unpublished
General Electric Company

Multicomputer processing in laboratory automation*

by C. E. KLOPFENSTEIN

University of Oregon
Eugene, Oregon

and

C. L. WILKINS

University of Nebraska
Lincoln, Nebraska

INTRODUCTION

Widespread availability of minicomputers in the scientific laboratory has become a reality in recent years. With the price and size reductions which have taken place, it is now practical to use these machines as routine tools in the laboratory environment. However, in order to effectively implement this new research tool the scientist has had to learn its limitations as well as its capabilities. It is well-known by computer scientists that, in general, the trade-off made by programmers is memory size *vs.* speed of execution. In other words, the more memory the programmer has available, the faster running program he may write and, conversely, the less memory he has, the slower will be the execution times. While there are a great many qualifications to this broad generalization, it still represents a recognizable truth.

Since memories available in the laboratory minicomputer are generally in the 4 to 12 K word range, the experimenter has had to painfully learn the lesson mentioned above. Attention to good programming practice has helped alleviate the problems of limited memory, but major problems still remain. Nowhere does this become more evident than in the program development stage. Traditionally, the minicomputer has employed multipass assembly procedures, usually utilizing a slow output device such as the teletype to produce program listings. In practice, it rapidly becomes apparent that such an assembly procedure is highly unsatisfactory in all but a very few cases. In this paper we will describe an alternate to this approach and a

number of techniques to facilitate program debugging using simulation methods. Also, the use of high level languages will be discussed in order to show how the usefulness of the minicomputer may be maximized by offsetting its limitations.

If we consider, first, the program assembly process, it is apparent that the rate-limiting step is input and output. The slow devices we are forced to use can easily insure that an assembly on a laboratory computer can take anywhere from several minutes to an hour or more. One solution would be to install high-speed peripheral devices (lineprinters, magnetic tapes, magnetic drums, disks, etc.) but, of course, these may easily cost more than the computer itself, so this approach is often unsatisfactory. Short of doing this, is there any way for the user to avoid being I/O limited in the critical program development and debugging phases? We believe the answer is yes. The following pages will describe software we have developed for one particular minicomputer, the Varian 620/i, to permit the programmer to use a much larger computer system for much of his program development. To achieve this end, both an assembler and simulator have been written (in FORTRAN) to run on the IBM 360 to assemble 620/i code, and to simulate the execution of the programs thus produced. Additionally, high level language compiler development is under way. These compilers will operate in an analogous fashion, producing machine programs which will then directly execute on the smaller computer. While we are well aware that the techniques mentioned above have been in use by computer designers and manufacturers for years, no such widespread use has been evident among those who have recently discovered the usefulness of the small computer as a laboratory tool. Accordingly, the end user rarely

* We gratefully acknowledge support of the National Science Foundation through grants GJ-441 and GJ-393.

specifies the sort of software mentioned above and, consequently, little such software has been provided by minicomputer producers.**

We began, about two years ago, the development of a program to train chemistry students, both graduate and undergraduate, in the use of the small computer in the chemistry laboratory. Accordingly, small computers especially configured for this purpose were acquired and employed in the early stages of course development. Since we were using 4K computers equipped with ASR 33 teletypes and fast paper tape I/O (300 cps read, 120 cps punch) for instructional purposes, we noted at the outset that our student programmers would often spend 45 to 60 minutes assembling a program and comparable times for performing the elementary debugging associated with syntax errors, misspunches, and the like. We estimated that as much as 90 percent of the computer time was being used for program development, with only about 10 percent being used for running the experiments in laboratory data acquisition and control which were our prime interest. This was clearly an unacceptable ratio and we immediately began to consider how we might make use of off-line (from the small computer) assembly and simulation techniques in order to improve the ratio and to free the laboratory computer for those tasks it does best.

THE USE OF DAS 360 AND SIM 620

Our solution was two programs, written in FORTRAN (DAS360 and SIM620) which would rapidly assemble and simulate 620/i code on the IBM 360. Now it became possible to provide far more adequate diagnostic messages (necessarily limited and cryptic in the Varian assembler) and to greatly facilitate the whole programming process. Students could now enter their programs *via* punched cards to the IBM 360, operating in the batch mode, and have a program assembled in a few seconds, as well as have a listing returned with comprehensive diagnostic messages. At the University of Oregon, where a research computer equipped with IBM compatible magnetic tape was available, the student programs were assembled on the 360, the assembled programs written on magnetic tape and the resulting tape transferred to the 620/i where the programs were dumped on paper tape and returned to the students along with the assembly listing for debugging or execution. At the University of Nebraska, students were permitted to use remote job entry CRT

terminals (first IBM 2260, later Bunker-Ramo 2206 terminals and the University of Nebraska Remote Operating System (NUROS) to enter their Varian 620/i programs first to a disk file, then directly into the job queue for assembly. Students had the option of placing the assembled programs together with their error messages on a disk file for immediate viewing on the terminals in the chemistry building, and obtaining a printed listing later or, alternately, obtaining a printed listing only. Punched card output of the assembled programs was available to all students. Both the assembler and simulator were stored on the 360 disk files and their execution could be invoked *via* a catalogued procedure in much the same way as a user would invoke any other program (*e.g.* the FORTRAN or COBOL compilers). Figure 1 contains a block diagram of the major components of the University of Nebraska system. In this way, it became possible to instruct far more students than could have conceivably been handled had we been restricted to the use of the small computer only. Furthermore, it was now possible for any student or faculty member at either of the Universities to make use of the programs, the only restriction being that they needed to have a valid account number to allow them to use the 360. Quite simple instructions for the use of both assembler and simulator were developed and freely distributed to any student who desired them.

The simulator (SIM620) proved to be at least as useful as the assembler for the program debugging

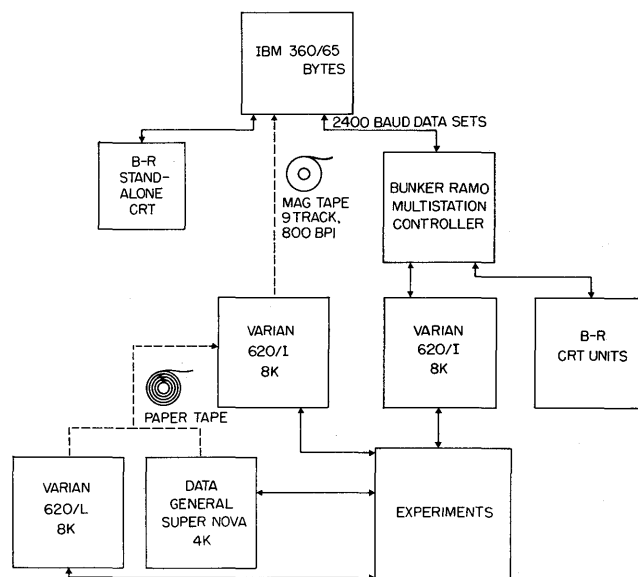


Figure 1—Block diagram, University of Nebraska distributed computer system for chemistry

** This situation has changed rapidly and now this software is increasingly available.

phases. As we mentioned earlier, it was our observation that, initially, as much as 90 percent of the mini-computer time was absorbed by the process of program debugging. Since those using the computer were either (a) beginning students who had never seen a computer before, (b) researchers writing code for a specific research application (many of whom had never seen a computer before, either), or (c) students and researchers engaged in the final check-out of completed programs, it was apparent that—short of buying many expensive peripherals or several more teaching computer systems—we were not going to be able to significantly change the debugging/experiment use ratio with laboratory computer systems. The obvious solution to the problem was to make use of the powerful, fast 360, insofar as possible. We therefore developed a program which allowed the larger computer to simulate the operation of programs assembled for use with the laboratory computer. This simulator also allows the programmer to enter limited amounts of data in order to test the operation of his 620/i program. The results of this simulation are quite useful and allow programs to be debugged very much faster and more efficiently than would otherwise be possible. A listing containing a full trace of program operation is produced. The contents of any or all operational registers before the execution of every instruction, the instructions executed and their memory locations may be included in this listing. At the University of Nebraska, this listing can be placed on magnetic disk files at the computer center and viewed by the student on the chemistry department CRT terminals. In this way, the student has, in effect, a remote 620/i to use for program debugging. The information thus obtained can be collected in a small fraction of the time it would otherwise take. Through use of the simulator, large numbers of students can simultaneously check programs for correct operation and only after they are reasonably certain their programs are error-free do they need to use the laboratory computer. This makes possible the restriction of its use to the types of laboratory problems most important to successful implementation of the system in the chemistry laboratory.

Once DAS360 and SIM620 had been successfully implemented, we turned our attention to improvement of debugging tools for that essential element of the experiments, the on-line testing phase. Particularly important for this is the availability of a fast effective means of interpretatively executing programs, changing and displaying core contents, and trapping certain conditions. With the 620/i, as with most small computers, utility programs to perform certain of these tasks were provided. In order to attain all the capa-

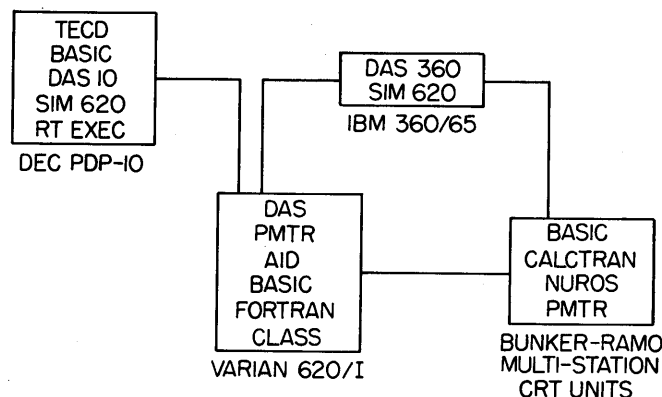


Figure 2—Interrelationships of software packages

bilities we required it was, however, necessary to extensively modify the software provided by the manufacturer. The resulting program (AIDF) is described below.

The basic premise behind the design of AIDF was that nearly all of the scientists and students who debug 620/I assembly programs can easily interpret machine code in actual format. Accordingly, a simulator-interpreter was prepared that provides complete program tracing and trapping facilities, with or without a teletype listing of the conditions of all registers before and after execution of each instruction. In this regard, listings similar to that provided by the 360 simulator are provided. However, since the programmer generally will use AIDF interactively, a much more flexible command set was made available. Provisions were made for listing and changing core from the teletype during execution of programs, as well as for simulating memory protection and features to pass control to command mode on execution of certain user defined "illegal" instructions. Also, the user can define certain unused operation codes to perform calls to his routines for fast non-interpretive execution. Multiply-Divide and other optional hardware are in this way emulated by the interpreter. To provide even more rapid debugging capabilities, all interpretive output may be directed to a storage oscilloscope. In this latter mode, execution rates of about 10 instructions per second with a full trace are attained. Using this debugging aid, students spend less wasted time at the computer console, and they have hard copy to carry away and survey at their leisure. Without AIDF, many fewer students would have access to the minicomputer for debugging assembly language programs. The interrelationships of these various software packages as well as an I/O translator (PMTR) and some others are summarized in Figure 2.

TABLE I—Basic Subroutines

CALL	FUNCTION
A. For Analog Input	
CALL SADC, G, C Dual slope integrating ADC 14-Bit—110 conv/sec Max	G=Gain 1=1, 2=8, 4=64, 8=512
CALL SADCF, C, S	C=Channel 1=1, 2=2, 4=3, 8=4
Succ. Approx ADC	S =Sense on DIO for block in start (0-7)
13 Bit—100 Kc conv/sec Max	
CALL RADC, V	Reads one value from DSIADC
CALL RADCB, S, N, A(I)	Reads N values at S points/sec into array A from DSIADC
CALL RADCT, V, T	Simultaneously reads one value from DSIADC and the timer
CALL RADCF, S, N, A(I)	Equal RADS except for fast ADC
B. For Analog Output	
CALL ODAX, C, V	Outputs V to DAC number C Channels 2 and 7 are 10 Bit DAC's Channels 4, 5, and 6 are 15 Bit DAC's
CALL ODAC, X, Y	Simultaneously outputs X and Y to Channels 2 and 7 and strokes 611 storage scope beam. Used for scope and X-Y re- corder drivers. (Range ± 500)
C. For Scope Control	
CALL SCOPE	All output goes to scope
CALL TELY	All output goes to teletype
CALL SIZE, S	Sets scope characters to size S Usual size is 4
CALL POS, X, Y	Causes next character to be put at X-Y on scope (Range ± 500)
CALL ERASE	Erases scope
D. X-Y Recorder Pen	
Control	
CALL PENU	Lifts pen and waits one second
CALL PEND	Puts pen down and waits one second
E. External Controls	
CALL EEXC, C	Strobes Channel C (0-7) on DIO
CALL EXCD, C, D	Strobes Channel C (0-7) on Device D (0-63)
F. Senses	
CALL ISEN, C, A	Returns A=1 if sense on Channel C of DIO is true, otherwise A=0

TABLE I—Continued

CALL	FUNCTION
G. Digital I/O	
CALL ODIO, W	Outputs W to lamps and digital drivers on DIO
CALL RDIO, C, W	Inputs from DIO to W from Channel C (0=16 Bit Buffer, 1=switches)
CALL RCTR, C, T, A	Reads the digital counter on Channel C (1 or 2) for time T (1 or 10 sec) into variable A
H. System Control	
CALL MAGT	Loads Magnetic Tape Operating System

HIGH LEVEL LANGUAGES

We will now examine the potential usage of high level languages with minicomputers. The advantages of providing laboratory users with high level language capabilities are multifold. The most important is that the time between conception and execution of an experiment is reduced to a minimum, which not only increases the effective usefulness of the minicomputer, but also greatly reduces the "activation barrier" to scientists who have never used the small computer into trying various experiments. Likewise, it becomes possible to incorporate experiments that illustrate data acquisition techniques into laboratory courses which are already established in the curriculum of a university or college.

One popular misconception held among those who have not previously used portable small computers (8K core with teletype I/O) is that high level languages such as BASIC and FORTRAN cannot be used for real-time applications due to their limited speed of execution and the large core storage space required for their use. However, in most experiments, the input data rate requirements are specified independently from the output requirements, and usually even the time between the input and output functions may be specified separately. Accordingly, for those applications where the timing requirements can be modularized, it seems reasonable to provide high level languages for coding the calculations, with assembly language sections or routines for input and output. Core storage remains a problem, but even with only 8K of core, programs with over 150 FORTRAN statements can be supported.

Provisions for calling subroutines are part of the languages FORTRAN and BASIC, and since these are the most commonly used compiler and interpretive languages today, we have provided a set of subroutines that allows the high level user access to virtually every

peripheral device required for real-time computing. Table I lists these routines. The devices include clocks, timers, analog-digital converters, digital-analog converters, relays, lamp indicators, and the like. Electrical connections are provided through a standard front panel using VICI or Banana plugs so that students can easily and rapidly change experiments without requiring even a screw driver. Our experience indicates that even relatively demanding tasks, such as low resolution mass spectrometry data processing, can have the calculation and teletype input-output code written in FORTRAN.

Student experimenters prefer to use the slower, but more interactive, Real Time Basic package. The most important feature of Real Time Basic is that programs can be written, tested, and modified interactively from an on-line teletype eliminating the compilation and link edit steps of FORTRAN. Statements are checked for correct syntax on loading rather than at run time to even further accelerate the rate of program development. The experiments we have developed for students to perform in BASIC range from analysis of gas chromatography data to complicated simulations in physical chemistry. A typical student-written program for acquiring and plotting 250 pairs of time and voltage readings (from a pH meter, in this case) is as follows:

```

90 DIM V(250),T(250)
100 CALL SADC,4,1
110 FOR I=1 TO 250
120 CALL RADCT,V(I),T(I)
130 CALL ODAC,V(I),T(I)/10.
140 NEXT I
    
```

The language CLASS (Computer Language for Spectroscopy Systems) was developed by Varian Associates to provide a simple programming medium to solve slow to medium speed on-line data collection and massaging problems. This string processing language provides an easy to learn system for instrument control and acquisition of lists of data—all arithmetic operations are performed in a double precision (31 bit) stack, and data may be stored in either single or double word formats. Programs are written as strings composed of macro calls which are executed interpretively. Any string or collection of strings can easily be defined by new macro names, or, old, unused macros may be deleted to free needed core space. All program generation steps occur interactively at the teletype immediately before an experiment.

We have used CLASS to support acquisition and processing of data from various spectrophotometers and electron spin resonance spectrometers.

The sample CLASS program given below demonstrates the code required to collect a 3000 point electron spin resonance spectrum in each of the possible pro-

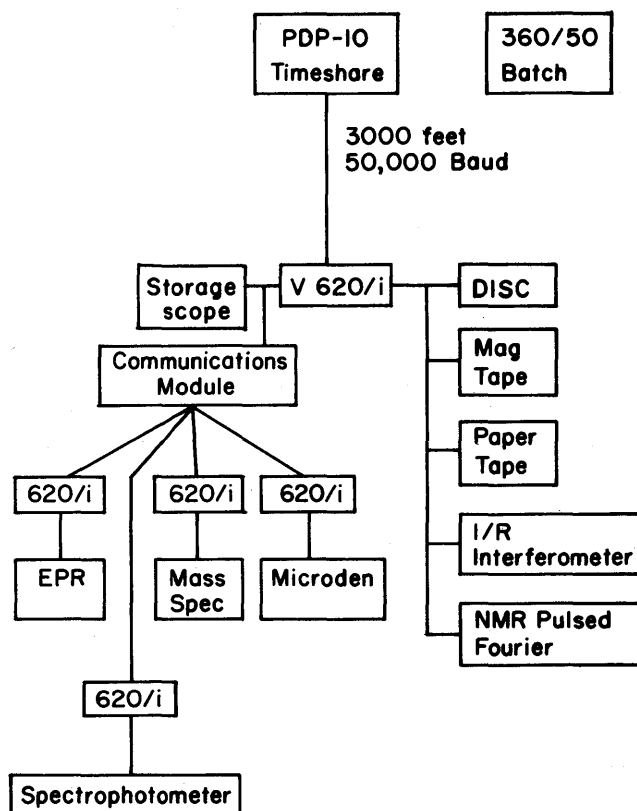


Figure 3—Block diagram, University of Oregon distributed computer system for chemistry

gramming modes. Effective use of the lowest string mode requires a complete understanding of the architecture of the language, whereas use of the highest level requires only instruction in the use of a teletype.

TTY	TTY	TTY
STRING	RETURN	EPR
R24	PEND	FILT 2
R77	END	DEST 100
•	COLLECT	STRT 1
R17	FILT 2	END 5999
C2	PUT	GO
R33	DEST 100	
A100	STRT 1	
C1	END 5999	
C 5999	RIGHT	
R 76	RIGHT	
R 76	END	
•	GO	
M		
GO		
Lowest "Assembly"	Simple Macro	Highest Macro
Level	Level	Level

On execution of the sample programs given above,

the recorder arm would first be moved to the extreme left with the pen up. After that, the pen would be lowered, three thousand data points collected (at a rate of one point every two deciseconds) and the data plotted. For each data point the recorder would be stepped two recorder increments to the right.

The maximum advantage in the use of linked computers is that expensive bulk storage capabilities need to be provided at only one location. At the University of Oregon, we are currently implementing a communications network that will support direct transmission of data between several Varian 620/i computers and a centrally located Digital Equipment Corporation PDP-10 computer. The tremendous volumes of data generated in gas chromatographic mass spectrometer and Fourier infrared experiments is collected with an

8K machine and transferred for computation to the PDP-10. The system hardware is outlined in Figure 3.

SUMMARY

In this paper we have discussed an approach to laboratory computing which allows the experimenter to take advantage of each of a variety of programming languages and hardware facilities. Through use of the distributed computing systems described, it has been possible to serve a wide range of users ranging from the inexperienced student to the experienced researcher, and to allow them to make use of laboratory computers routinely.

Enhancement of chemical measurement techniques by real-time computer interaction.

by SAM P. PERONE

Purdue University
Lafayette, Indiana

The small, dedicated, laboratory computer can provide enhanced capability for electroanalytical measurement techniques in the chemistry laboratory. The work described here is based primarily on three recent publications by Perone, Jones, and Gutknecht.^{1,2,3} The particular electrochemical analysis techniques to which computerization has been applied are stationary electrode polarography (SEP),⁴ and the closely-related derivative voltammetry.^{5,6} However, the principles and methodology described should be generally applicable to other electroanalytical techniques, and, perhaps, to chemical experimentation, in general.

Certainly, one very important way in which the on-line digital computer can improve the capabilities of chemical measurement techniques is simply to provide automated experimentation, data assimilation, and straightforward data processing. This has been demonstrated amply already for electroanalytical instrumentation.^{7,8,9} However, these approaches are bounded ultimately by the inherent limitations of the particular measurement technique. To take full advantage of the dedicated computer, one should utilize its capabilities for rapid "intelligent" feedback and incorporate the computer into the experimental control loop, as shown in Figure 1. Thus, the computer could monitor the experiment; process the data as the experiment progresses; and, "intelligently" modify the course of the experiment to provide optimum measurement conditions. Of course, the "intelligence" is related to the programming skill and the experimental intuition of the programmer. Moreover, the transfer function for "intelligent" response will depend on the degree of sophistication and number of calculations and decisions which must be made in "real-time"—*i.e.*, between successively acquired data points; also important are the computer's speed, and hardware arithmetic and logical capabilities. (A more quantitative discussion of response factors is given below.)

The important feature of this latter approach—real-

time computer optimization of analytical measurements—is that a measurement technique can be generated, which would be unattainable without the aid of an on-line computer. Moreover, in the process of developing this application, the experimenter must necessarily investigate systematically those experimental parameters which most critically determine the computer optimization of the measurement. This can be done only with the computer in the control loop. However, the results of these investigations then provide the foundation for the design of new instrumental methods which might be implemented independent of an on-line computer. These aspects of laboratory computer applications—real-time computer interaction with instrumentation, and computerized experimental design of interactive instrumentation—will be discussed below.

RESPONSE FACTORS FOR REAL-TIME COMPUTER INTERACTION

The response of the computerized control loop depicted in Figure 1 can be quantitatively evaluated. The analysis is similar in many respects to that applied in characterizing analog operational amplifier response.¹⁰ That is, one can describe a *transfer function* for the digital computer control element. This transfer function can be defined as the dependence of *computer power* on *stimulus frequency*.

To define these terms, consider that a digital computer is a programmable device which executes arithmetic, logical, and input/output operations in sequential fashion. (References 9, 11, and 12 may provide useful background on computer programming for laboratory applications.) Thus, computer power is directly proportional to *execution time available*. It is also directly related to inherent hardware capabilities, such as instruction execution time, micro-programming

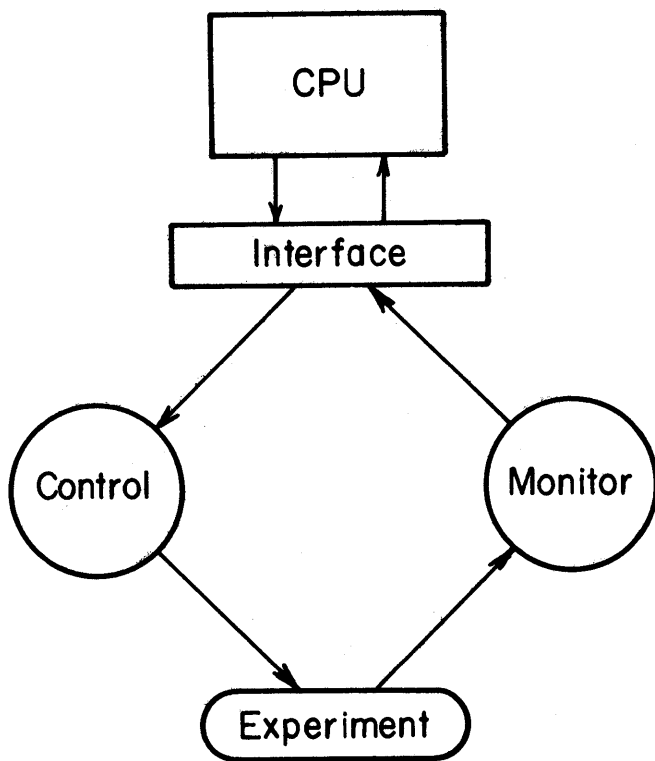


Figure 1—Block diagram of laboratory instrumentation with computerized feedback loop

characteristics, input/output structure, etc. However, for a given computer and a specific experimental application, the critical variable is execution time available.

The stimulus frequency, f , is simply the frequency at which the computer is “poked” by the experiment requesting some external service. In the simplest situation, f is the data acquisition frequency—the rate at which digitized data are made available to the computer from the experiment.

Consider now the role played by the on-line computer during the execution of a given experiment. In some experiments, the computer’s service, as each datum is made available from the digital data acquisition system, may involve only inputting the datum, saving it in memory, and some bookkeeping. Typical *service time*, τ , may be 20 or 30 μsec . In a more complex case, where some computational evaluation of the data, logical decisions, and possible experimental control operations may be required also in real-time, τ may be considerably longer. This latter case is the type with which we are concerned here.

Now we can define a transfer function for a computerized system. The *real-time computer power*, P , can be equated to the available computational time

between stimuli as given in Equation 1,

$$P = 1/f - \tau = 1 - f\tau/f \tag{1}$$

where P is in units of time. The dependence of P on f and τ is shown graphically in Figure 2. The smallest value of P shown on the ordinate axis in Figure 2 is 1.0 μsec .

The type of response analysis presented above is admittedly simple-minded. Nevertheless, it can be very useful in the design and implementation of computer-interactive instrumentation. Certain related factors must be considered, however. First of all, the discussions here relate only to *dedicated* systems, where the computer is interfaced to a single instrument. Thus, it is assumed that the execution of the real-time service program begins “immediately” upon request from the interfaced instrument. In fact, there is some *minimum response time*, τ_0 , required. The magnitude of τ_0 depends on the computer hardware features and also depends on the programmer’s choice of response mechanism. If he chooses to use the computer’s *Interrupt System*, τ_0 may be as short as a microsecond. If he chooses to use *program-controlled response*, where the computer is programmed to sit in a loop waiting for a service request, τ_0 may be several microseconds.

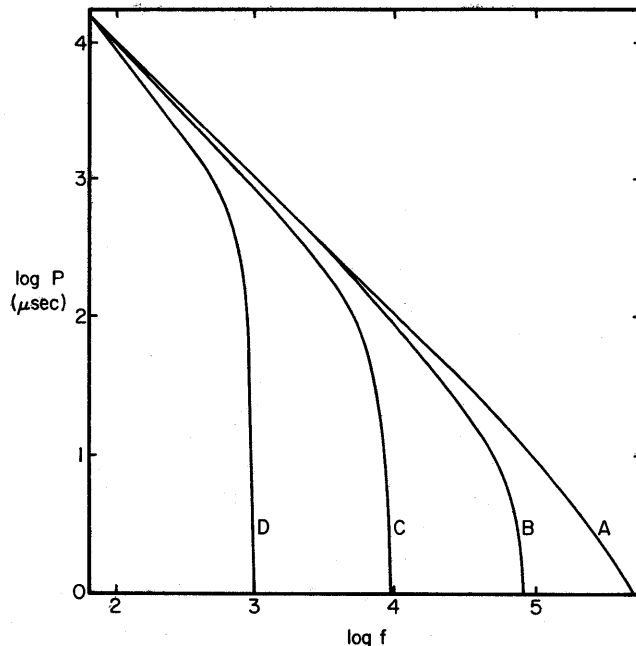


Figure 2—Dependence of real-time computational power on stimulus frequency

- A. $\tau = 10^{-6}$ sec
- B. $\tau = 10^{-5}$ sec
- C. $\tau = 10^{-4}$ sec
- D. $\tau = 10^{-3}$ sec

A second assumption is that the computer is the slowest element in the control loop. That is, it is assumed that all analog instrumentation controlled or measured by the computer does not limit the overall system response.

Another relevant consideration is that the value for service time, τ , used in any calculations should be the "worst case" value. That is, where alternative program pathways exist, assume that conditions will always require the longest path.

SOME SAMPLE CALCULATIONS

Consider now how one might use the system response analysis described above for the design of a particular experimental application involving real-time computer interaction. First, the essential elements of the minimal experimental service program—including the required input/output and bookkeeping instructions to be executed for each stimulus—must be established. The time required for these operations plus the minimum response time, τ_0 , correspond to the *minimum service time*, τ_M . The transfer function for this value of τ_M establishes the real-time computer time available, τ_A , where

$$\tau_A = P(f, \tau_M) \quad (2)$$

Thus, the programmer can calculate τ_A for the specific data acquisition or service frequency, f , required in his application. Then, he must establish τ_R , the *real-time computational time* required to provide the desired calculations, decisions, and experimental control operations to allow computer interaction with the experiment. The linear combination of the two programming segments results in a *total real-time service time*, τ_T , where

$$\tau_T = \tau_M + \tau_R \quad (3)$$

If the programming is such that $\tau_R \leq \tau_A$, the proposed application is feasible. Alternatively, for a given value of τ_T , one can calculate the maximum data acquisition frequency allowed. This can be obtained by setting $P=0$. Then, $1/f = 1/f_{\max} = \tau_T$.

A specific example should illustrate the above discussion. One computer system used by this author has a basic machine cycle time of 1.6 μsec , and each instruction is some integral multiple of this value. For a particular application, program-controlled service was used requiring a worst-case response, τ_0 , of 4.8 μsec . The additional minimal service programming required 8 cycles, 12.8 μsec . Thus, τ_M was 17.6 μsec . The desired data acquisition frequency, f , was 10KHz. Therefore, the value of τ_A was computed from Equations 1 and 2 to be 82.4 μsec . Thus, the real-time interactive pro-

gramming had to have a worst-case execution time, τ_R , less than 82.4 μsec . This allowed programming requiring no more than 51 machine cycles.

The following discussion presents a specific application of real-time computer programming for the enhancement of stationary electrode polarographic measurement capabilities. The work described is taken from Reference 1 and illustrates the implementation of principles discussed above.

REAL-TIME COMPUTER CONTROL IN STATIONARY ELECTRODE POLAROGRAPHY

Stationary Electrode Polarography, (SEP), is a chemical analysis technique where solutions are analyzed by the measurement of electrolysis currents that flow when the cell voltage is swept. Despite its many desirable characteristics for automated analysis,⁷ SEP is limited seriously for application to mixtures. Because of the continuous nature of the experiment, currents from easily-reducible species continue to flow and contribute to, distort, or mask currents measured for more-difficultly-reducible species. The non-ideal aspect of the technique is the fact that a *continuous* linearly varying potential is applied to the electrolysis cell, regardless of the composition of the sample. If the linear sweep were *discontinuous*, stopping briefly after each reduction step to allow the more complete dissipation of the easily-reducible species in the diffusion layer around the electrode, the interference with reduction steps for more difficultly-reducible species would be considerably diminished. However, such a discontinuous or "interrupted-sweep" experiment would require some foreknowledge as to the composition of the mixture—and this is not a likely situation in real analytical situations.

The work of Perone, Jones, and Gutknecht¹ illustrated how one can take advantage of the control capabilities of the on-line digital computer to overcome the resolution problems of stationary electrode polarography. The approach taken was to allow the computer to interact with the experiment in real-time to generate an interrupted-sweep experiment which was effectively "sample-oriented". Some details of that work will be presented here.

Sample-oriented analysis—Interrupted sweep approach

Resolution limits

The theories of conventional stationary electrode polarography⁴ and stationary electrode polarography with derivative read-out^{5,6} allow the accurate prediction

TABLE I—Values of Current Function Ratios, $\chi(at)_p/\chi(at)_E$, $\chi'(at)_p/\chi'(at)_E$, and $\chi''(at)_p/\chi''(at)_E$ as Functions of $(E - E_{1/2})^a$

$n(E - E_{1/2}), \text{mV}$	$\chi(at)_p/\chi(at)_E$	$\chi'(at)_p/\chi'(at)_E$	$\chi''(at)_p/\chi''(at)_E$
-150	1.813	6.044	15.01
-175	1.991	7.983	24.08
-200	2.153	10.18	35.03
-225	2.299	12.54	50.26
-250	2.436	15.02	68.00
-275	2.563	17.63	88.90
-300	2.671	20.22	115.6
-325	2.788	22.99	144.5
-350	2.896	26.16	172.6
-400	3.096	32.28	241.0
-450	3.304	38.90	330.0

^aSee References 4 and 5 for definition of symbols.

of resolution limits. This can be done by calculating the theoretical ratio of the current functions at the peak and at some potential beyond the peak. Using the mathematical approaches outlined previously,⁴⁻⁶ and considering only reversible systems, this has been done for 0-, 1st-, and 2nd-derivative measurements.¹³ The results are shown in Table I. [The peaks chosen for the 1st- and 2nd-derivative current functions are the largest ones in each case—at $n(E - E_{1/2}) = 18.8$ and -14.4 mV, respectively.] Note that the interference with a succeeding reduction diminishes considerably with derivative measurements and with increased potential separation of reduction steps; resolution increases with the order of the derivative; however, even with the derivative measurement, the resolution is not particularly good when reduction steps are closer together than about $300/n$ mV.

Thus, considering arbitrarily a separation of $300/n$ mV and equal n - and D -values, it would be possible to resolve, with 5 percent contribution of the first reduction step to the second, concentration ratios of 1:7, 1:1, and 5.8:1 for the 0-, 1st-, and 2nd-derivative measurements, respectively. These calculations are exclusive of any other background contributions, with the assumption that they are negligible or can be measured independently for correction.

It would, of course, be possible to correct mathematically for the interference caused by overlapping reduction steps. However, a limit is reached with this approach when the interference is so great as to preclude even the recognition of a succeeding reduction step. In any event, this approach has been considered² and will be discussed later.

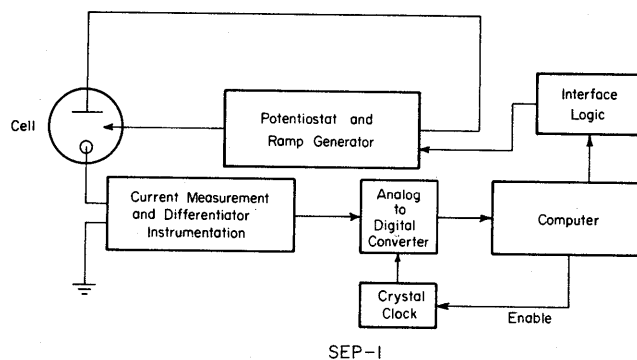


Figure 3—System block diagram

Details of Interrupted-Sweep Experiment

The interrupted-sweep experiment involves a computer-controlled potentiostat (described in Reference 1), and real-time analysis of fast-sweep derivative polarographic data. (A system block diagram is shown in Figure 3.) The computer continuously monitors the experimental output, is instantaneously aware of the occurrence of reduction steps, and can interrupt the linear potential sweep at an appropriate potential cathodic of each peak. The interrupt potential is held for a length of time computed to allow sufficient depletion of the electroactive species in the diffusion

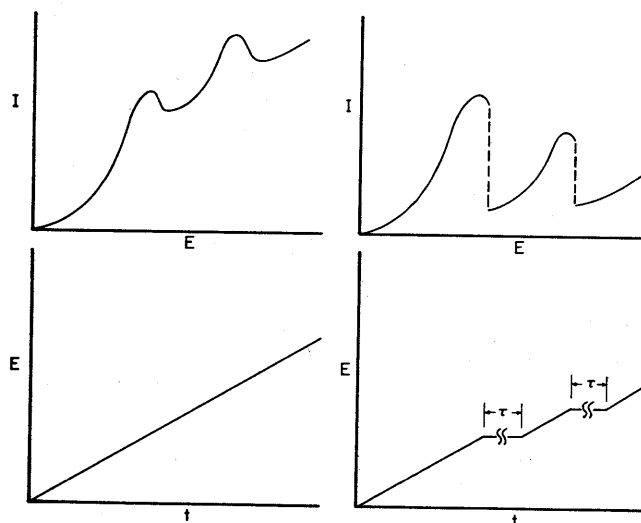
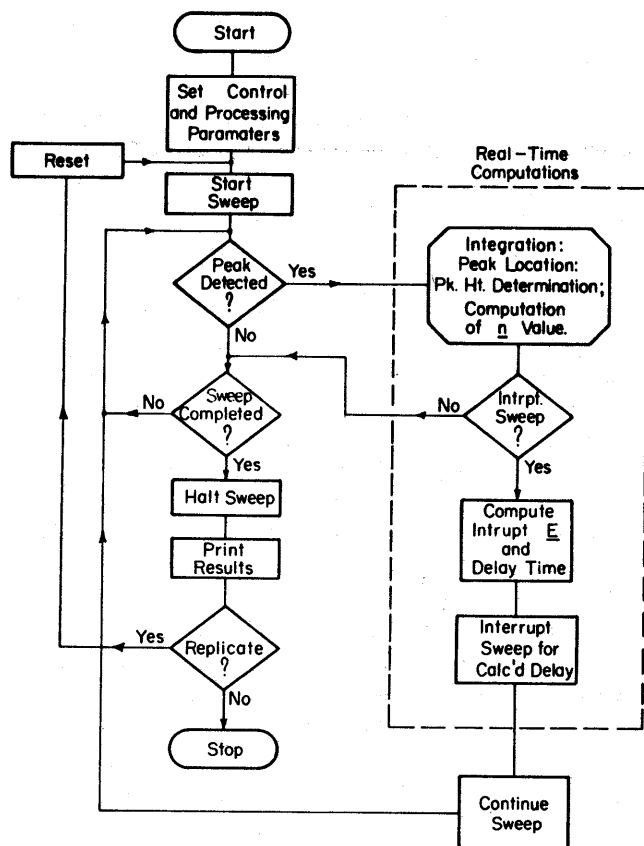


Figure 4—Comparison of normal and interrupted-sweep stationary electrode polarography

- Stationary electrode polarogram (W/O interrupt)
- Stationary electrode polarogram (With interrupt)
- Applied cell potential (W/O interrupt)
- Applied cell potential (With interrupt)

layer, and then the sweep is restarted. The interrupt delay time, τ' , is calculated in proportion to the magnitude of the reduction step, with the restriction that τ' not be so long as to cause convection processes to occur or to allow significant electrolysis of the next electroactive species. Thus, the controlling potential function—which is basically a series of ramp-and-hold steps—will be different for each experiment, depending on the sample mixture and composition. The experiment is custom-tailored to the sample—*i.e.*, *sample-oriented*. A simplified comparison of the continuous- and interrupted-sweep experiments is shown in Figure 4. A flowchart of the computer-controlled experiment is given in Figure 5. The computer used in this work was a Hewlett-Packard Model 2115A with 16-bit word size, 8,192-word core memory, 2.0 μsec cycle time, and a hardware extended arithmetic unit. The data acquisition system included a 10-bit, 33 μsec conversion-time analog-to-digital converter (ADC). The timing was provided by an external 10 MHz crystal clock scaled down to 1 KHz, which was the maximum data acquisition frequency, f ,



Flowchart for Real-Time Computer-Optimized S.E.P.

Figure 5—Flowchart for real-time computer-optimized SEP

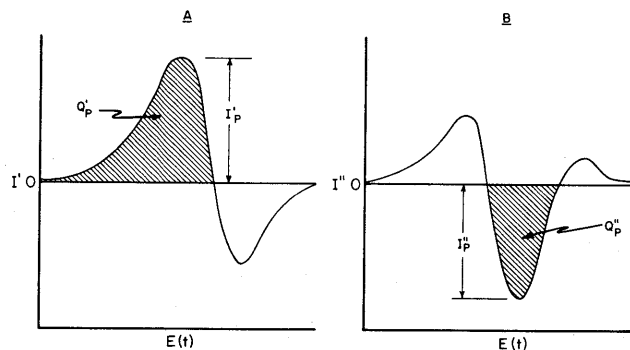


Figure 6—Analytical measurements from first- and second-derivative voltametric data
A. First derivative
B. Second derivative

used for all experiments. A detailed discussion of interfacing and control logic is given in Reference 1.

The information taken in by the computer is extracted from either the 1st- or 2nd-derivative signal. Only the negative region of the derivative signal is seen by the analog-to-digital converter, and the measuring circuit is arranged so that only the largest peak in each case is the correct polarity. The result is shown in Figure 6. The computer is programmed to look for sharp peaks—above an arbitrary threshold—and to measure and store peak heights, peak areas, and peak locations in real-time.

When a complete peak is observed—*i.e.*, one which goes above threshold, goes through a sharp maximum, and then comes back down below threshold—the computer executes the maximum real-time programming, calculating the n -value, the potential at which the sweep should be interrupted, E_D , and the delay time, τ' . (The maximum total real-time programming time, τ_T , is about 800 μsec .) Then the computer allows the sweep to continue (if necessary) until the desired interrupt potential (E_D) is reached; the sweep is interrupted at this point for time, τ' ; the sweep is then reinitiated with the computer looking for the next reduction step, ready to reexecute a similar interrupted-sweep. (The delay time, τ' , is computed in proportion to the peak height, with the restriction that τ' be between 100 and 1000 ms.)

Advantages of real-time calculations

Uses of Integral Data

It was possible to integrate the derivative peaks (Figure 6) seen by the computer in real-time and to use

TABLE II—Interrupt Potential Required for Specified Surface Ratio, C_o/C_R

$(E_d - E_{1/2})$ C_o/C_R	0 1/1	-28.5/ n 1/3	-59.1/ n 1/10	-100/ n 1/50	-118/ n 1/100	-177/ n 1/1000
--------------------------------	----------	-------------------	--------------------	-------------------	--------------------	---------------------

these integral data for analytical purposes, for calculating appropriate interrupt potentials, and to provide diagnostic information. The area under a peak is directly proportional to the peak height and, therefore, to the concentration of electroactive species. Thus, the peak integral, Q_p , is a concentration-dependent output. Moreover, should the peak location routine in the program fail because the derivative peaks are too noisy, broad, or small, the peak integrals will still be taken and provide a useful, reliable, source of analytical information. In addition, a peak area threshold is incorporated into the program, whereby the area of a given peak must exceed some arbitrary value before the computer will recognize a signal excursion as a *bona fide* reduction peak. This is a very useful processing parameter.

In the case of 1st-derivative read-out, the value of the integral of the observed peak is equivalent to the peak height of the conventional stationary electrode polarogram. It has been shown previously that, for a reversible system, the ratio of the 1st-derivative peak height, I'_p , to the conventional peak height, i_p , is related to n^6 , as given by Equation 4,

$$I'_p/i_p = 13.2nv \quad (4)$$

where v is the scan rate in volts/sec. Thus, a determination of the ratio of the derivative peak height to the peak integral can lead to an evaluation of n , and the computer is programmed to do this in real-time so that the information is available for interrupted-sweep decisions. This information is also useful for qualitative identification of species or for providing an error diagnostic if wrong n -values are obtained for known systems.

A similar relationship exists between the n -value and the ratio of the 2nd-derivative peak height to the peak integral. The measured integral is equivalent to the difference between the positive- and negative-going 1st-derivative peaks. The relationship can be calculated from previous theoretical data⁵ and is given in Equation 5. (Note the error in Equation 2 of Reference 1.)

$$I''_p/(I'_{p1} - I'_{p2}) = 21.2nv \quad (5)$$

Thus, the n -value can be obtained from either the 1st- or 2nd-derivative measurements. For reversible processes, the computed n -value is accurate. For irreversible processes, the n -value at least reflects the broadness of

the peak, and this is useful for the interrupt potential calculations discussed below.

Selection of interrupt potential

In the interrupted-sweep experiment, the potential, E_D , selected to be held during the delay period, τ' , is critical. The objective is to select a potential which is cathodic enough to deplete adequately the electroactive species in the diffusion layer. That is, the potential should be chosen such that the concentration ratio C_o/C_R , approaches zero at the electrode surface.

The obvious problem is that selecting a value of E_D cathodic enough to truly deplete the electroactive species at the electrode would eliminate the possibility of observing a succeeding closely-spaced reduction. Thus, a compromise must be reached, and a knowledge of the n -value for the reduction step on which the delay is made is useful in selecting the appropriate interrupt potential.

In this work, interrupted-sweep experiments were run with the interrupt potential (E_D) selected by the computer after it has observed a complete derivative peak—*i.e.*, when the derivative signal is going through zero. E_D is selected relative to the potential, E_Z , at which the derivative signal goes through zero. The computer uses information provided initially by the operator in order to calculate $E_D - E_Z$. That is, the operator initially specifies $n(E_D - E_Z)$; the computer determines n and E_Z , and then selects E_D . Experiments are reported below where the operator-selected value of $n(E_D - E_Z)$ was varied for a series of runs to observe the effect of E_D on quantitative resolution.

The influence of E_D on the surface concentrations of species in the redox couple, O and R , is shown in Table II. The calculations for Table II are based on the Nernst equation and reversible behavior. Also, it should be noted that $E_Z - E_{1/2}$ is $-28.5/n$ mV for the 1st-derivative measurement, and $-66.5/n$ mV for the 2nd-derivative measurement.

Results

Two different two-component systems were studied in this work. The first system consisted of Tl(I) and Pb(II) in 1.0M NaOH electrolyte. The half-wave potentials for these two species are separated by

approximately 280 mV. The second system studied was that of Pb(II) and Cd(II) in a 2M ammonium acetate-acetic acid electrolyte. The $E_{1/2}$ separation for this case was approximately 150 mV. All runs were made at a scan rate of 1.00 V/sec. Data points were taken at 1- or 2-mV intervals, and both the first- and second-derivatives of the reduction currents were observed for each system.

Various experiments were applied to the two systems. These included normal SEP and interrupted-sweep SEP with computer-selected interrupt potential. The values of $(E_D - E_Z)$ employed varied from 0/n mV to values which resulted in noticeable charging spike interference. The delay time, τ' , for the first peak in each example was always near to or equal to 1000 msec., since the first peaks were always quite large. The results of these tests are summarized in Table III. Also included in these tables are the theoretical estimates of overlap error based on the data from Table I.

TABLE III—Peak Derivative Measurements of Smaller Component with and without Computer Interaction for Binary Mixtures.

Mixture	Condition	$E_D - E_Z$, mV	% Std., I_p'	% Std., I_p''
30:1 Tl:Pb	w/o interrupt	—	50.8 ^a	92.6 ^b
"	interrupted-sweep	0/n	89.9	100.8
"	"	-20/n	92.8	100.8
"	"	-100/n	97.9	—
10:1 Pb:Cd	w/o interrupt	—	45.3 ^c	91.9 ^d
"	interrupted-sweep	0/n	104.4	100.0
"	"	-20/n	104.8	86.9
"	"	-30/n	102.3	—
100:1 Tl:Pb	w/o interrupt	—	no peak ^e detected	88.7 ^f
"	interrupted-sweep	-40/n	70.8	101.0
"	"	-100/n	83.5	102.9
1000:1 Tl:Pb	w/o interrupt	—	no peak ^g detected	no peak ^h detected
"	interrupted-sweep	-40/n	"	78
"	"	-100/n	"	98

Predicted errors: (a) -48.4%, (b) -5.2%, (c) -45.6%, (d) -8.2%, (e) -160%, (f) -17%, (g) -1600%, (h) -170%. (Predicted errors based on Table I and known concentration ratios.)

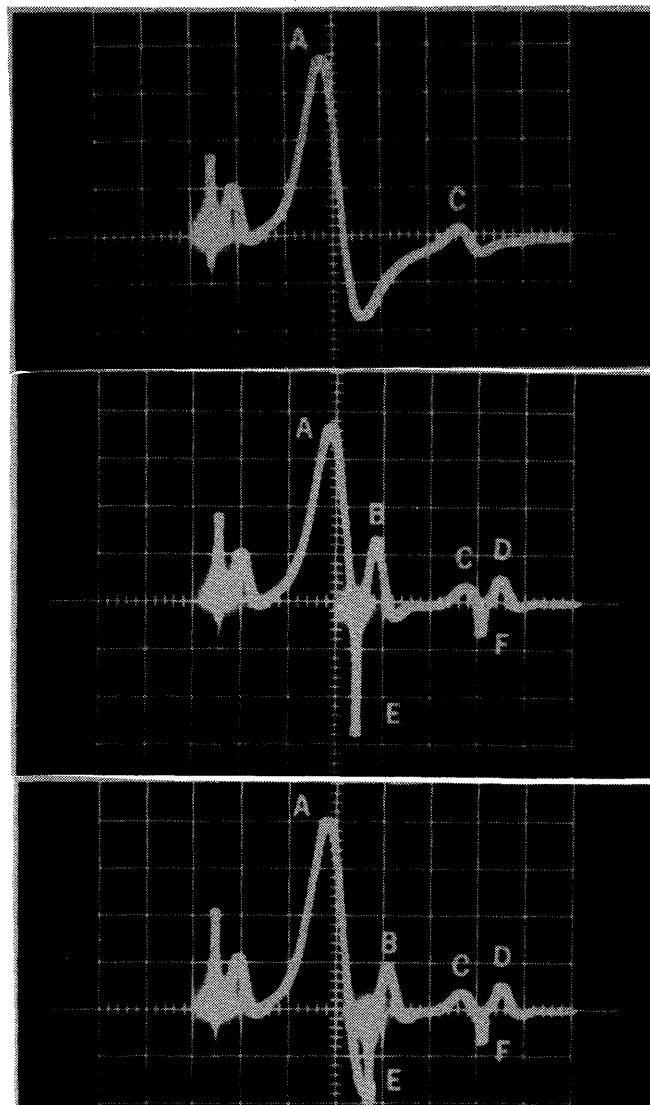


Figure 7—First-derivative curves for 30:1 [Tl(I)]-[Pb(II)] system

$3.21 \times 10^{-4}M$ Tl(I), $1.05 \times 10^{-6}M$ Pb(II),
1.0M NaOH

Upper trace: W/O interrupt

Middle trace: With interrupt; $E_D - E_Z = -10/n$ mV

Lower trace: With interrupt; $E_D - E_Z = -40/n$ mV

The signals observed are: (A) Tl(I) signal; (C) Pb(II) signal; (B) and (D) charging spikes; and (E) and (F) current decay occurring during interrupt.

The 1st-derivative data for the [Tl(I)]-[Pb(II)] system show continued improvement with increasing $(E_D - E_Z)$. However, beyond $(E_D - E_Z) = -100/n$ mV, some distortion apparently is caused by the charging spike of the restarted sweep overlapping slightly with the Pb(II) signal.

The error due to overlapping reduction signals shown in the 2nd-derivative data for the [Tl(I)]-[Pb(II)] system is small even without the interrupted-sweep. (This is predicted, of course, from Table I.) The

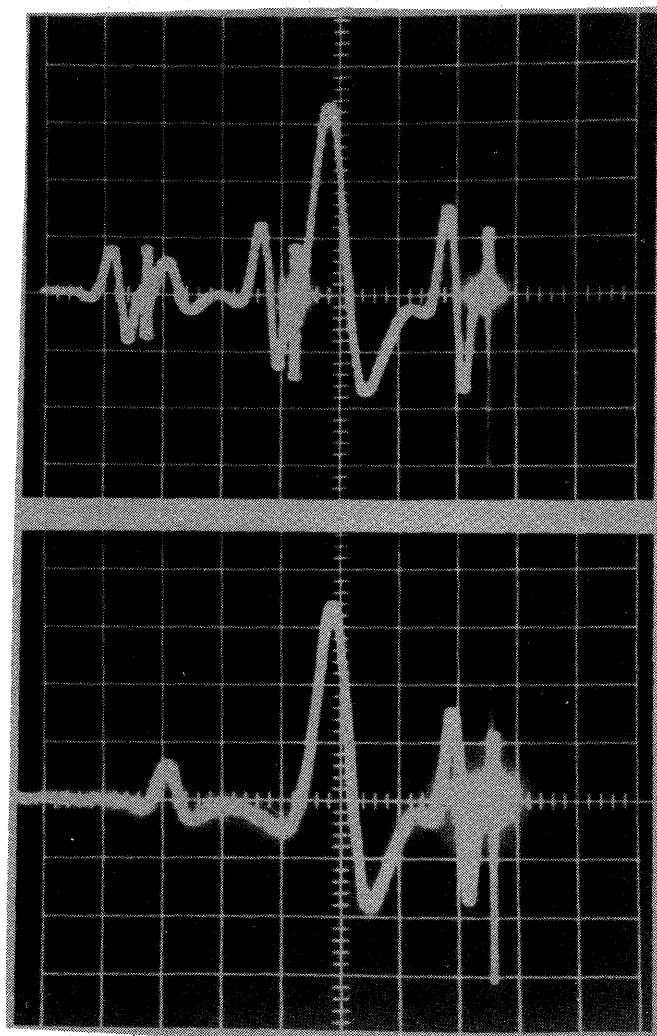


Figure 8—Second-derivative curves for 30:1 [Tl(I)]-[Pb(II)] system
 $3.21 \times 10^{-4} M$ Tl(I), $1.05 \times 10^{-5} M$ Pb(II),
 $1.0 M$ NaOH
 Upper trace: W/O interrupt
 Lower trace: With interrupt; $E_D - E_Z = -10/n$ mV

improvement with the interrupted-sweep is significant, however. The experimental effects of the interrupted-sweep for both the first- and second-derivatives can be visualized in Figures 7 and 8.

The first-derivative data for the [Pb(II)]-[Cd(II)] system, using the interrupted-sweep experiment, show some contribution to the Cd(II) peak from the charging spike, even with small values of $(E_D - E_Z)$. This interference is illustrated in Figure 9, and results in a small positive error for the second peak.

The width of the charging current spike for both the first- and second-derivative is observed to be approxi-

mately 100 mV. Thus, if E_D is to be set cathodic of E_Z , the end of the first signal peak and the start of the second signal peak must be at least 100 mV apart. This is a significant limitation on the interrupted-sweep experiment.

The second-derivative data for the [Pb(II)]-[Cd(II)] system show better overall results than the first-derivative data. This was to be expected, on the basis of earlier studies,^{5,7} and the data of Table I. It was observed that for both systems the peak integral data, Q'_p and Q''_p , show greater error than the peak

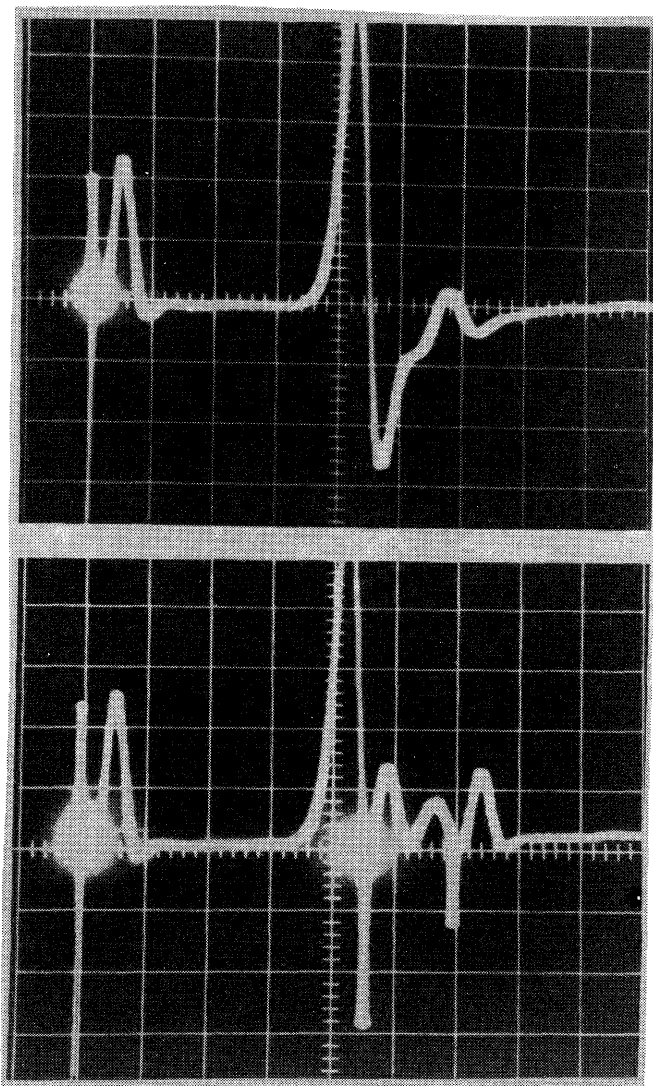


Figure 9—First-derivative curves for 10:1 [Pb(II)]-[Cd(II)] system
 $1.05 \times 10^{-4} M$ Pb(II), $1.11 \times 10^{-5} M$ Cd(II),
 $2 M$ NH_4OAc , $2 M$ HOAc
 Upper trace: W/O interrupt
 Lower trace: With interrupt; $E_D - E_Z = 0/n$ mV

height data. This is not unexpected, because the area from the peak base is missed when peak overlap occurs. Thus, peak areas should not be the primary source of analytical data.

The results for $[Tl(I)]-[Pb(II)]$ 100:1 and 1000:1 mixtures in 1.0M NaOH show a considerable improvement in quantitative resolution when the interrupted-

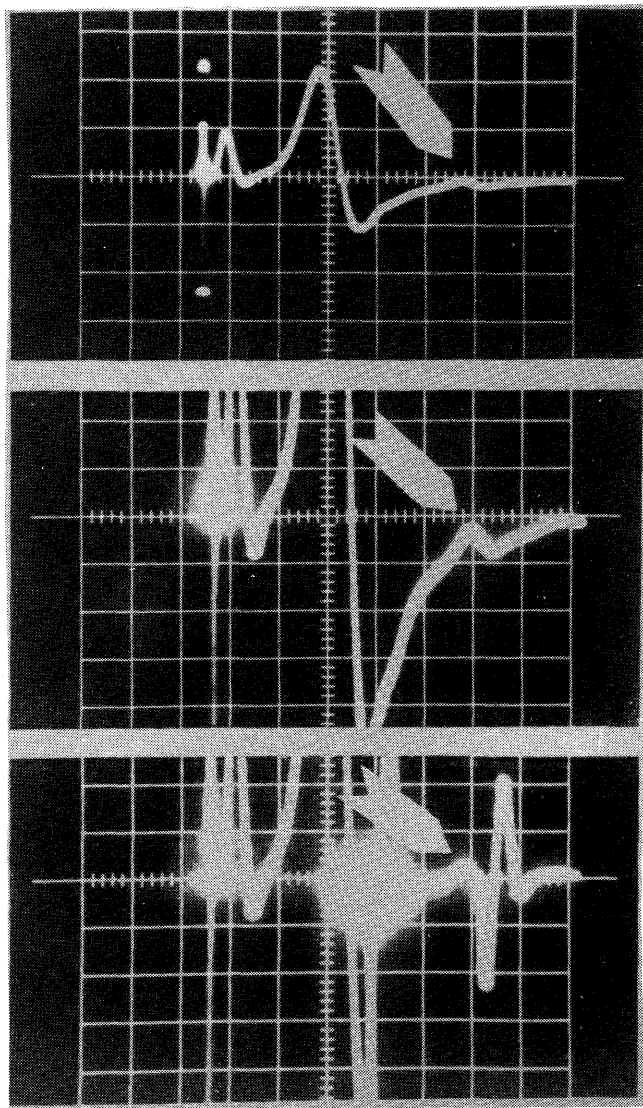


Figure 10—First-derivative curves for 100:1 $[Tl(I)]-[Pb(II)]$ system
 $2.68 \times 10^{-4}M$ Tl(I), $2.63 \times 10^{-6}M$ Pb(II),
 1.0M NaOH
 (Arrow shows Pb(II) peak)
 Upper trace: W/O interrupt
 Middle trace: W/O interrupt; sensitivity increased 5×
 Lower trace: With interrupt, $E_D - E_Z = -40/n$ mV; sensitivity same as middle trace

sweep experiment is employed. For the 100:1 mixture, the second peak is undetectable with a 1st-derivative read-out (see Figure 10). With the interrupted-sweep, however, not only is the second peak detectable, but the measured value comes up to 84 percent of the correct value, for 1st-derivative read-out, and 100 percent for 2nd-derivative read-out. For the 1000:1 mixture the second peak is undetectable, even with a 2nd-derivative measurement. Employing the interrupted-sweep experiment with 2nd-derivative read-out, however, reliable and quantitative detection could be obtained.

Observations

The work described here was intended to demonstrate that an on-line digital computer could be used to optimize an experimental measurement technique by real-time interaction with the experiment. The results clearly show a dramatic improvement in quantitative resolution of overlapping reduction signals, provided a minimum $E_{1,2}$ separation of about 150 mV is present. Thus, the optimized measurement is subject to at least this one severe limitation; but, nevertheless, appears quite useful. Most importantly, the principle of real-time computer-optimized measurements in electroanalysis was demonstrated by application to a real system.

COMPUTERIZED EXPERIMENTAL DESIGN OF INTERACTIVE INSTRUMENTATION

The experimental method described above illustrates the application of an on-line digital computer to generate, evaluate, and optimize a new electroanalytical approach. The general-purpose laboratory computer is well-suited for this task because of the ease with which programmed control functions can be modified during the development of experimental control characteristics. However, having arrived at an optimum set of experimental control features the continued dedicated use of an on-line computer for routine application of the technique might not be economically feasible. Thus, a more practical approach should be taken to adapt the technique developed with the general-purpose computer system for routine laboratory application. In another publication,³ Jones and Perone described the incorporation of the optimum parameters determined from the earlier work,¹ summarized above, into a specialized instrument designed to generate the interrupted-sweep experiment without the need for an on-line computer. This later work³ demonstrated the value of the com-

puter-controlled experimentation for the *design* of interactive experimental techniques that can then be hardware-implemented. It also demonstrated that many programmed control operations can be converted easily to hardware logic and analog functions by using medium-scale integrated circuit (MSI) modules.¹⁴

The optimum parameters selected from the earlier work were incorporated into a device in which few manual operations were needed and which, as a result, would implement the technique for most, but not all, of the cases studied. Detailed descriptions of the instrumentation and approach are presented in Reference 3. Essentially the same functions as in the computerized technique were provided, but some changes were made. Only the 2nd-derivative peak height, I''_p , was used to extract quantitative information. The first-derivative zero crossing E'_z was used for qualitative identification, rather than the peak center of the 1st- or 2nd-derivative which was used in the computer-controlled technique. This was because of the ease with which the zero crossing could be detected with a hardware comparator. The peak width of the second derivative was used as an indication of n^5 , rather than the ratio of the peak height to its integral as in the earlier work. This was because the 2nd-derivative peak-width measurement was much easier to accomplish with hardware and gave very reproducible results. The interrupt time delay was proportional to the second-derivative peak height, as it was for the computerized approach, and was again limited to between 100 and 1000 msec. Because the earlier work had shown that adequate resolution could be obtained up to at least 100:1 mixtures for $E_D = E_Z$, no provision was made in the hardware device for adjusting $E_D - E_Z$ in normal automated operation.

When the hardware instrumentation approach³ was compared with the computerized experimentation of the earlier work,¹ several observations were made. First, the analytical results and limitations were essentially identical for the two approaches. The only exception was that the hardware instrumentation failed for a 1000:1 mixture. This was because the hardware device used a slightly less sensitive—though more reliable—peak detection method. Also, quantitative resolution of 1000:1 mixtures would require minimum peak separations of about $250/n$ mV and a value of $E_D - E_Z$ of $-100/n$ mV¹. Because the hardware device was designed for completely automated operation, E_D was set to always equal E_Z , and this allowed application to any system where peaks were separated by 150 mV or greater, up to peak ratios somewhat greater than 100:1.

The limitations mentioned above reflect the objective of hardware development. The device was built to handle most analytical situations with a minimum of

operator manipulations. Moreover, the cost was only about five percent of the computerized system. The device is much simpler to operate than the computerized instrumentation because only the initial cell potential, amplifier gain, and sweep mode (with or without sweep interrupt) need to be specified by the operator before an experiment. Nevertheless, it provides truly interactive instrumentation. Although the feedback is less flexible or "intelligent" than in the computerized system, it is optimized. In addition, the hardware device offers one other distinct advantage over the computerized approach. The sweep rate of the computerized system was limited by the time necessary to perform the real-time calculations. This amounted to about 800 μ sec between data points, which limited the data rate to about 1 KHz. If data are taken at each mV during the sweep, this limited the sweep rate to about 1V/sec. In the hardware device, this limitation does not exist as the hardware can perform several logical, arithmetic, storage, and control operations in parallel; thus, less real-time steps are required. With proper modifications to the potentiostat, current follower, differentiators, and filters, the interrupted-sweep experiment could be run at sweep rates of 100 V/second. With appropriate scaling of τ' , this would allow much shorter experimental times and might be useful for the analysis of unstable systems.

The most important point here, though, is that the hardware instrumentation could not have been designed readily without the prior investigative study¹ using the

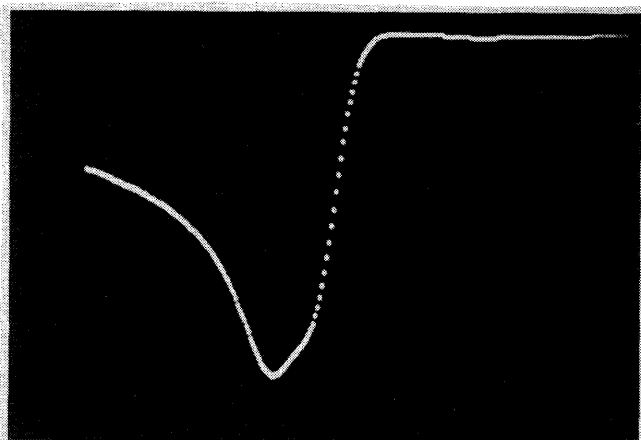


Figure 11—SEP curves for Real 1:1 [In(III)]-[Cd(II)]system
Peak Potential Separations 48 mv.
 $3.84 \times 10^{-5} M$ In(III), $3.95 \times 10^{-5} M$ Cd(II), 1.0M HCl
Voltage range shown: $-0.300 \rightarrow -0.800$ v. vs.
S.C.E.
Maximum peak current: 3.7 μ a.

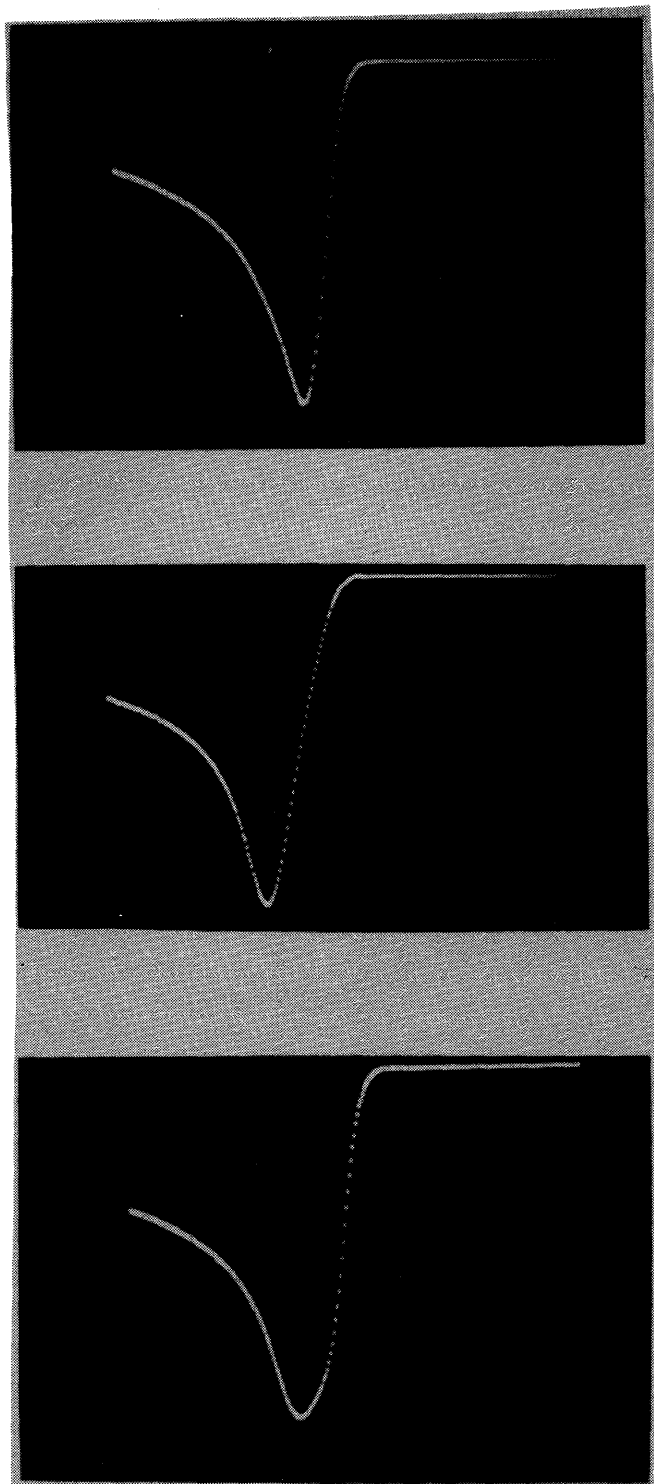


Figure 12—SEP curves for synthetic 1:1, 1:5, and 5:1 mixtures of [In(III)]—[Cd(II)]

Peak potential separations 38–42 mv.

Voltage range shown: $-0.300 \rightarrow -0.800$ v. vs. S.C.E.

on-line general-purpose computer. The optimum design parameters—as well as the very feasibility—of the interrupted-sweep technique were evaluated with the computerized experimentation.

COMPUTERIZED RESOLUTION OF CLOSELY-SPACED PEAKS IN STATIONARY ELECTRODE POLAROGRAPHY

The experimental work described above demonstrated how computerized instrumental interaction could improve the quantitative resolution capabilities of SEP. However, that approach fails for peaks separated by less than about 150 mV. An alternative approach must be used to handle electroanalytical samples where SEP peaks are more severely overlapped. One approach taken has been presented by Gutknecht and Perone.²

The approach involved extracting the analytical information from SEP data using mathematical deconvolution techniques. An empirical equation was developed which describes the general stationary electrode polarogram for a wide variety of electroactive species. The function is fit to a number of *standard* polarograms, and the constants of the function, as specifically determined for each species, are stored in computer memory. Upon analysis of an unknown mixture, these constants are used to regenerate the standard curves, a composite of which is then fit to the unknown signal. In the fitting process, account is taken of overlap distortion as well as experimental fluctuation of peak potentials.

The small computer was used to perform several different functions in the development of the on-line electroanalytical system. Primary among these were experimental control, timing, synchronization, and data acquisition functions. In addition, with the aid of an oscilloscopic display system, off-line simulation studies were carried out to evaluate empirical equations developed for later on-line data processing. Finally, the computer was used to process SEP data acquired on-line for qualitative and quantitative information. The processing approach proved especially valuable for

Upper trace: $4.80 \times 10^{-5} M$ In(III), $4.94 \times 10^{-5} M$ Cd(II), 1.0M HCl

Maximum peak current: 4.8 μa .

Middle trace: $0.960 \times 10^{-5} M$ In(III), $4.94 \times 10^{-5} M$ Cd(II), 1.0M HCl

Maximum peak current: 3.0 μa .

Lower trace: $4.80 \times 10^{-5} M$ In(III), $0.988 \times 10^{-5} M$ Cd(II), 1.0M HCl

Maximum peak current: 3.5 μa .

the analysis of mixtures of similar concentrations where the overlap was so severe as to preclude visible recognition of the individual signals.

A discussion of the numerical deconvolution approach is beyond the scope of this paper, and the reader is referred to the original work for details.² However, a brief summary of the results of that work can be presented.

It was shown that for mixtures of similar concentrations of In(III) and Cd(II) in 1.0M HCl, with a peak potential separation of 48 mV, it was possible to detect and quantitatively resolve the overlapped peaks with relative errors the order of 1 to 2 percent. A polarographic trace for a 1:1 mixture is shown in Figure 11. The approach was applicable to mixtures with concentration ratios as great as 10:1.

To establish the limiting peak separation which could be handled by the deconvolution technique, synthetically generated polarograms of In(III) - Cd(II) mixtures were used where peak separations were varied. The limiting peak separation was found to be about 40 mV. Mixtures with concentration ratios as great as 5:1 could be qualitatively identified and quantitatively resolved with about one to two percent relative errors. By contrast, simple simultaneous equation calculations led to relative errors the order of 10 to 35 percent. Moreover, the visual detection of the two individual peaks was not possible, as shown in Figure 12.

It should be obvious that the numerical deconvolution approach and the computerized interaction approach are complementary in many ways. The latter is applicable to widely-spaced peaks with large concentration ratios; the former is applicable to very closely-spaced peaks, but can handle more limited peak ratios. Both approaches represent a significant enhancement of measurement capabilities in stationary electrode polarography.

CONCLUSIONS

Certain observations should be made here. First of all, real-time computer interaction with experimentation is not the answer to all measurement problems. In some cases, in fact, it makes the problem worse. For example, the interrupted-sweep approach is completely inappropriate for SEP measurements of closely-spaced reduction peaks. One is tempted to generalize and state that the interactive approach fails when the interaction distorts the fundamental processes of interest. It was shown here how one might use the numerical analysis capabilities of the small computer for solution of these measurement problems.

A second point is that one may not need to devise a real-time interaction scheme to achieve computerized optimization of experimental measurements. A perfectly adequate approach might involve an iterative method where the computer is programmed to analyze the data from a completed experimental run; make decisions regarding modification of controlled parameters for improved measurements; and then reinitiate the experiment under new conditions.

A third point to be made here is that, as demonstrated above, it may not be necessary to require a digital computer for implementation of real-time interactive instrumentation. However, the investigation of the approach and the establishment of the optimum mode of interaction are greatly facilitated by the on-line digital computer. Subsequent hardware implementation of the approach can be straightforward and economical.

A final observation to be made here is to attempt to define in general the experimental situations where real-time computerized interaction is advantageous and/or necessary for optimization of measurements. These situations seem to include those where separate dynamic experiment-associated chemical or physical processes occur which interfere with the measurement of interest at a particular time during the experiment. If the interfering processes can be independently evaluated by real-time computations, computerized interaction may be advantageous. If post-mortem analysis of unoptimized experimental measurements does not provide adequate information for subsequent experimental modifications, real-time computer interaction may be necessary for optimization.

It would be presumptuous on the part of this author to describe specifically how other measurement techniques might be optimized by real-time computer methods. Only the experienced worker skilled in the particular analytical method has the appropriate understanding and intuition for proper experimental design. However, several analytical methods have been recognized as being amenable to optimization by real-time computer measurements. These include gas chromatography,¹⁵ kinetic and other clinical methods of analysis,^{16,17} as well as coulometric analysis.¹⁸ Undoubtedly, many such applications will be developed in the near future.

ACKNOWLEDGMENTS

The support of the National Science Foundation, Grants No. GP-8677 and GP-21111, is gratefully acknowledged.

REFERENCES

- 1 S P PERONE D O JONES W F GUTKNECHT
Analytical chemistry Vol 41 1154 1969
- 2 W F GUTKNECHT S P PERONE
Analytical chemistry Vol 42 906 1970
- 3 D O JONES S P PERONE
Analytical chemistry Vol 42 1151 1970
- 4 R S NICHOLSON I SHAIN
Analytical chemistry Vol 36 706 1964
- 5 S P PERONE T R MUELLER
Analytical chemistry Vol 37 2 1965
- 6 C V EVINS S P PERONE
Analytical chemistry Vol 39 309 1967
- 7 S P PERONE J E HARRAR F B STEPHENS
R E ANDERSON
Analytical chemistry Vol 40 899 1968
- 8 G LAUER R ABEL F C ANSON
Analytical chemistry Vol 39 765 1967
- 9 G LAUER R A OSTERYOUNG
Analytical chemistry Vol 40 30A 1968
- 10 *Handbook of operational amplifier applications*
Burr-Brown Research Corporation
Tucson Arizona 1963
- 11 *Introduction to programming*
Digital Equipment Corporation
Maynard Massachusetts 1969
- 12 S P PERONE
Journal of chromatographic science Vol 7 714 1969
- 13 P E REINBOLD
MS thesis Department of Chemistry
Purdue University Lafayette Indiana 1968
- 14 J S SPRINGER
Analytical chemistry Vol 42 23A 1970
- 15 R G THURMAN K A MUELLER M F BURKE
Journal of chromatographic science Vol 9 77 1971
- 16 G E JAMES H L PARDUE
Analytical chemistry Vol 41 1618 1969
- 17 G P HICKS A A EGGERT E C TOREN JR
Analytical chemistry Vol 42 729 1970
- 18 F B STEPHENS F JAKOB L P RIGDON
J E HARRAR
Analytical chemistry Vol 42 764 1970

The television/computer system—The acquisition and processing of cardiac catheterization data using a small computer*

by H. DOMINIC J. COVVEY, ALLAN G. ADELMAN, CLARENCE H. FELDERHOF,
PAUL MENDLER, E. D. WIGLE and KENNETH W. TAYLOR

Toronto General Hospital
Toronto, Ontario, Canada

INTRODUCTION

One of the prime objectives of cardiovascular research is to assess the functional state of the heart especially the left ventricle, its main pumping chamber. The functional state of the left ventricle is determined by its dimensions,¹ volume,² the velocity of wall movement,³ the intracavity pressure,⁴ and the wall tension and stress.^{5,6,7,8}

This paper describes a semi-automated technique for obtaining parameters which indicate the functional state of the heart from left ventricular cineangiograms (35 mm. X-ray cine films of the heart taken while injecting a contrast material into the pumping chamber) and simultaneously recorded intracardiac pressures. Doing the measurements necessary to obtain function data is tedious and time consuming as dimensions must be measured from each frame of a cineangiogram of the left ventricle taken at 60 frames per second over several seconds, and correlated with the instantaneous cavity pressure. Displaying resultant function parameters in an intelligent fashion is also critical if they are to be useful to the clinician.

Two years ago the Cardiovascular Laboratory at Toronto General Hospital undertook to quantitatively assess the functional state of the human left ventricle. The resources available to this department dictated that any application of data processing equipment be modest, and that the equipment be housed within existing space. We, therefore, chose a small computer, optimizing on both the low cost and expandability of a local on-line system and the availability of software, peripherals and interfacing electronics. This equipment

in conjunction with a standard broadcast television system provides a powerful data acquisition and processing facility which occupies one office in the unit and has the capacity to deal with the large volume of multi-formatted data (digital, analog, pictorial, patient records) resultant from heart investigation procedures.

The total cost of the system has been about \$100,000 for hardware and \$20,000 for personnel. It is being used for research and development and is more sophisticated than necessary for routine work. For routine work a simpler configuration can be used. For example, a minimum system might include: a PDP-8/E (\$5,000), a teletype (\$1,700), disc or tape (\$8,000–\$10,000) interfacing (\$2,000–\$4,000), and some basic television equipment (\$5,000), or not more than \$27,000.

Other techniques have been developed for obtaining this information. They range from totally automated border recognition and dimension extraction systems^{9,10,11} to hand measurements.^{12,13} Between these extremes, there are: (a) semi-automated systems similar to ours,^{14,16} (b) a light pen system¹⁵ and (c) various techniques for obtaining border coordinates by standard X-Y position digitizers or scanners.^{17,18,19,20}

Totally automated systems involve long setup procedures, the digitization of entire pictures, and the presentation of this data to a large computer for analysis. This is often slow and extremely expensive. On the other hand, manual measurements are tedious, time consuming and the resultant data often requires some machine processing.

Three systems similar to ours²³ have been developed. Two use a dimension measuring interface similar to the one described in the text and illustrated in Figures 3–7: (1) the Bugwatcher,²¹ which uses a much more expensive computer, is very similar to ours in design, but is not used for cardiovascular work, and (2) an analog

* Work supported by the Ontario Heart Foundation and Toronto General Hospital

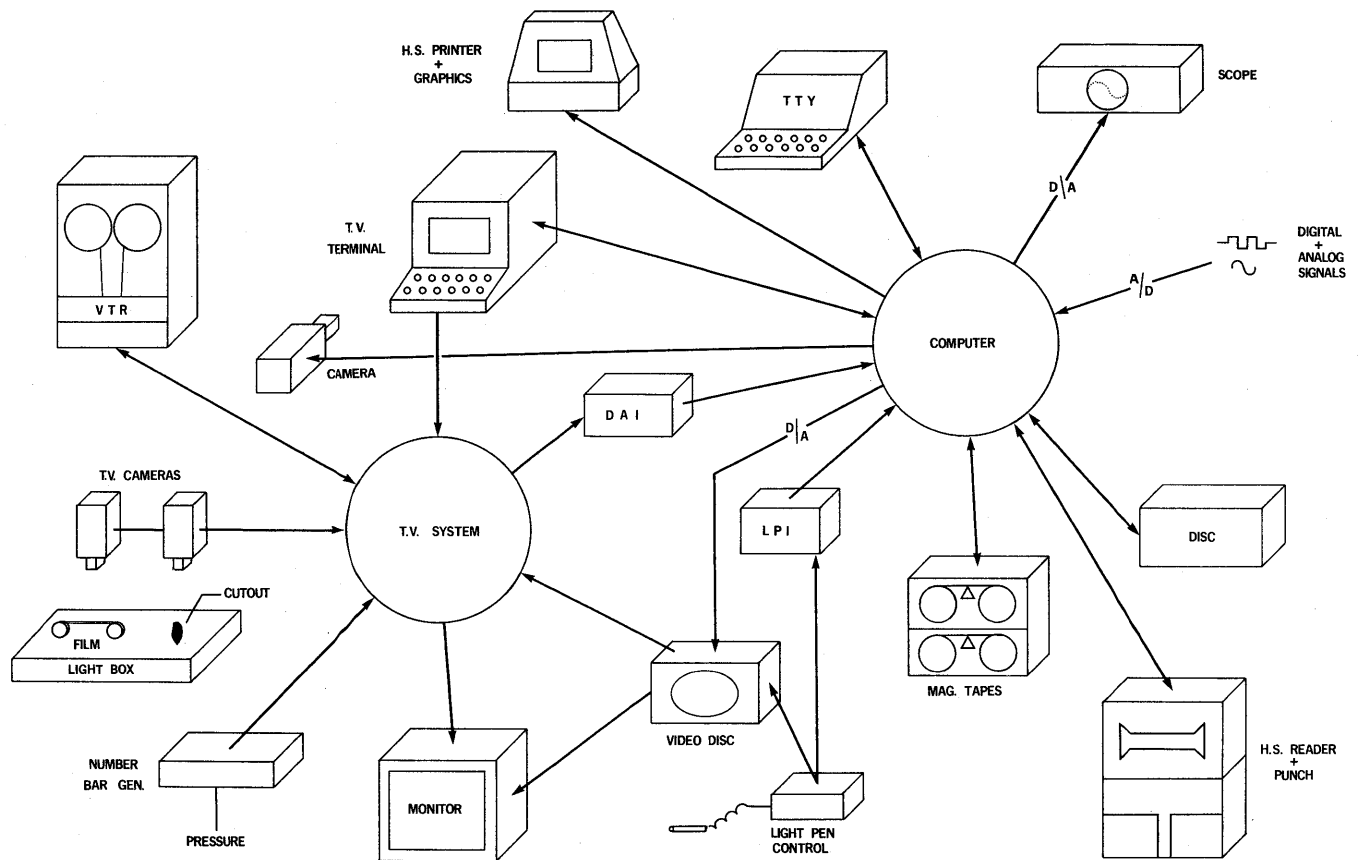


Figure 1—Diagram of the Television/Computer System. The computer (DEC PDP-8/I) can transfer data to or receive it from a number of peripherals (see Table II, text). The television system is composed of a master synchronization generator and sync and video-distribution circuits (see Table I, text for details). In addition, there are the interfaces which connect the computer and television systems. These include: the dimensional analysis interface (DAI), the light pen and light pen interface (LPI) and the analog-digital (A/D and D/A) converters. Certain peripherals are shared by both systems, permitting easy communication between them.

system,²² which, although real-time operation appears possible, suffers the limitation of having to depend on an expensive video magnetic disc and of making only area, length and one left ventricular width available as data. The third¹⁵ uses a light pen similar to that described here, but employs a scan converter and storage oscilloscope instead of a magnetic disc recorder for refreshing the display, and is less flexible in use.

THE TELEVISION/COMPUTER SYSTEM

The system we have designed (Figure 1) is based upon interfacing a television system with a small computer. The television system (Table I) (standard 525 line broadcast equipment), is inexpensive but very flexible. We use television as a brightness/voltage, dimension/time converter for pictures. In addition, available television circuits permit a number of useful

functions, e.g., selecting parts of a picture for examination, changing the quality of a picture, superimposing windows or other signals on a picture, recording televised signals, and presenting calculated or plotted data.

The main features of this work are the interfaces which permit the analog signals from the television system to be converted into digital format for input to the computer and allowing the computer, in turn, to communicate with the television system. The interfaces

TABLE I—The T.V. System

The main sub-elements of the Television System are:

1. Monitors (CONRAC, SONY, H/P)
2. Number and bar generating circuits
3. Television cameras (SHIBADEN)
4. Magnetic Tape (SONY 2")
5. The Vista 1 H
6. The Video Disc

TABLE II—The Computer System

The computer peripherals available to the 4K PDP-8/I are:

1. Magnetic tape (DEC, TU-55)
2. 32K Magnetic Disc (DEC, DF-32)
3. A/D, D/A converters, relay drivers, pulse inputs (DEC, AX08)
4. Oscilloscope display (TEKTRONIX, RM 503)
5. Teletype (ASR-33)
6. 180 LPM printer and graphic output terminal (LEIGH, ALPHAGRAPHIC)
7. 4800 Baud CRT terminal (INFOTON, VISTA 1 H)
8. Video disc via the AX08 (COLORADO VIDEO, VIDEO PLOTTER)

ensure that the dimension data measured from each television line in a picture are identified with that line, and are presented to the computer at an acceptable rate.

The computer, a 4K PDP-8/I (recently upgraded to 8K) has available to it a number of standard peripherals (Table II) for inputting and outputting the data and for combining dimension data with other measurements during catheterization.

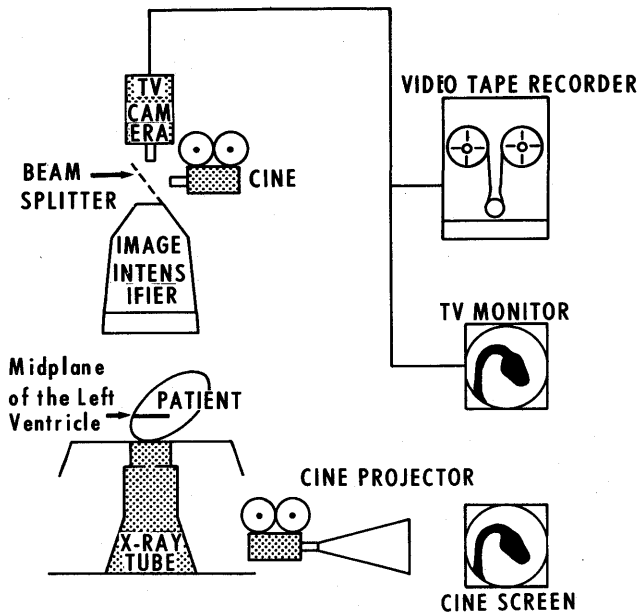


Figure 2—A diagram of the cineangiographic system used in these studies. The X-ray image of the left ventricle is brightened electronically and a 35 mm. cine camera photographs the images at 60 frames per second. The image is also relayed via a television camera to a television monitor and a video taperecorder. The cine film is later projected by a 35 mm. projector onto a screen where the left ventricular silhouette can be outlined. If the spatial relationships of the various interfaces are kept constant, the only variable affecting magnification in the system is the midplane of the left ventricle.

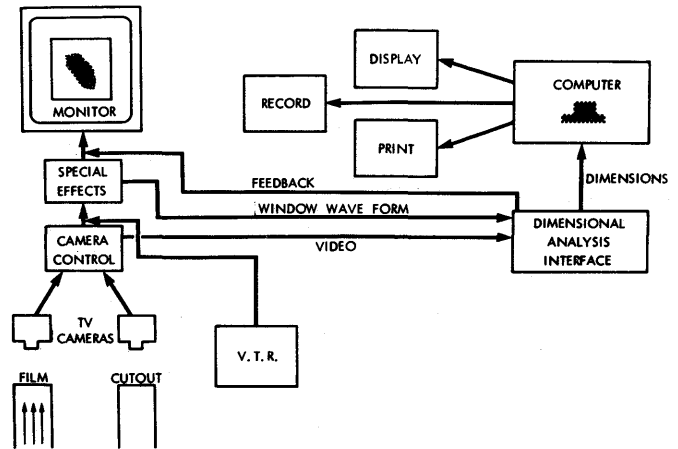


Figure 3—Cutouts, and pictures from film or videotape can be measured by the DAI. The video signal from these pictures is adjusted by means of the camera control and presented to the DAI for analysis. A special effects window, which allows the manual selection of a portion of the picture for analysis, is mixed with the video signal and appears on the monitor. The switching waveform from this special effects window is input to the DAI. The DAI can feed back the measured image to the monitor and transfer the measurements to the computer which can display, record or print the dimensions.

Presently, these measurements are made after the films obtained during cardiac catheterization are processed and returned to the unit. This entails a delay of about 24 hours before the catheterization data is available. However, ultimately our aim is to obtain data directly from video tape recordings of the angiogram and to play back the results into the television system during the catheterization.^{22,23}

CARDIAC CATHETERIZATION

Cardiac catheterization is done by introducing catheters (small, 2 mm. O.D., tubes) into the peripheral arteries or veins and advancing these into the heart chambers. The catheters are used to record pressure or to inject radio-opaque material to obtain high-contrast serial X-ray pictures of the chambers of the heart. In left ventricular function studies, one catheter is used for injecting contrast material and one for monitoring pressure. The main data obtained from this procedure are: (a) sequential films of the left ventricle (left ventricular cineangiograms) and (b) the analog left ventricular pressure recording.

In our work the angiograms have been recorded on 35 mm. cine film at 60 frames per second directly from an image intensifier (Figure 2). However, it is also possible to process full-size films from a high-speed film

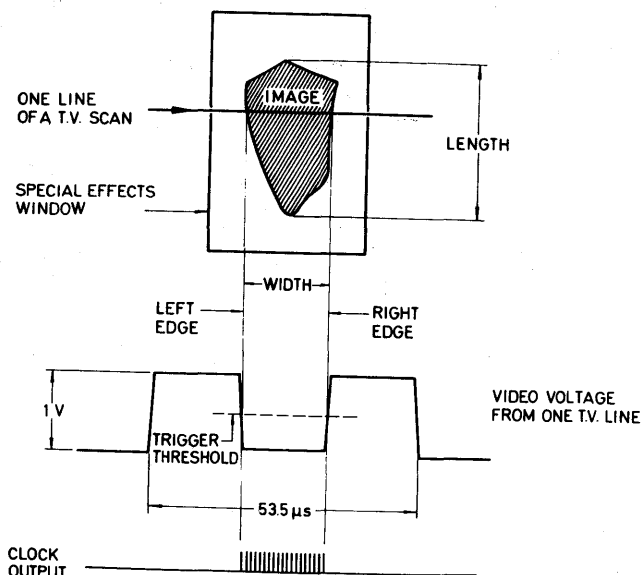


Figure 4—Schematic diagram showing how measurements are obtained from a television picture. When a line of the T.V. scan crosses the left edge of a dark image, a fall in the video voltage below a Schmitt trigger threshold starts a 10MHz clock, when the line crosses the right edge of the image the rise in the video voltage stops the clock. The number of clock pulses is proportional to the width of the image and the number of television lines crossing the image is proportional to its length. The threshold can be adjusted to define the edge of the image and a special effects window allows manual selection of the area of the picture which contains the image to be measured.

changer for greater detail or 16 mm. cine film of the televised X-ray image (kinescope recording) when television format would be advantageous for presenting other data on the frame. The pressure is recorded on photosensitive paper in an Electronics for Medicine oscillographic recorder, and also directly on film.

LEFT VENTRICULAR FUNCTION DATA

Dimension data

The Dimensional Analysis Interface (DAI)

Dimensions are measured from 35 mm. cine film projected onto paper by a stop frame projector (Tage Arno). A technician outlines the ventricle in each frame, draws axes on the outlines, cuts them out and puts them one-by-one on a light box where they are

viewed by a television camera. Alternatively, dimensions may be measured directly from each 35 mm. cine frame (Figure 6). The results obtained directly from film are less consistent than those obtained from cut-outs. However, with model studies,²⁵ the differences between measurements from film and cutouts were not significant.

The television image is input to the Dimensional Analysis Interface (DAI). The dimensional analysis interface measures the width of the image on each television line by a threshold technique and (Figure 4) makes the width and corresponding line number available to the computer. The way this is done is shown in detail in Figures 3-6. Programs calibrate the measured widths and length (proportional to the number of television lines crossing the left ventricular image) and calculate scaled widths, the area, the length and volume. The latter is calculated assuming that the left ventricle is circular in latitudinal cross-section.²⁶ The raw data is output on paper or magnetic tape (Figure 7). The tape record is later processed by programs which compute wall tension, stress, velocity and plot any selected dimension data.

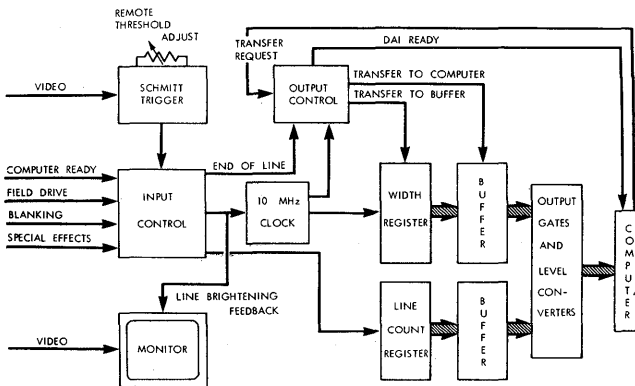


Figure 5—Schematic diagram of the DAI. Voltage transitions in the video signal turn on or off a Schmitt trigger which has an adjustable threshold. The output of the trigger is gated with the signals from the computer, with the television FIELD drive and BLANKING signals, and with the switching waveform from a SPECIAL EFFECTS generator. When these input conditions are satisfied, the time the trigger is on for each television line is measured by the number of clock pulses accumulated in a "width" register and is fed back to the television monitor as a bright line (Figure 6). The line number is recorded in a second register. At the end of each line during which the clock was on, the output control transfers the width and line registers to their respective buffers and indicates to the computer that it is ready for a TRANSFER REQUEST. When a TRANSFER REQUEST is received by the output control, the contents of the buffers are copied into the computer accumulator through a series of output gates and level converters.

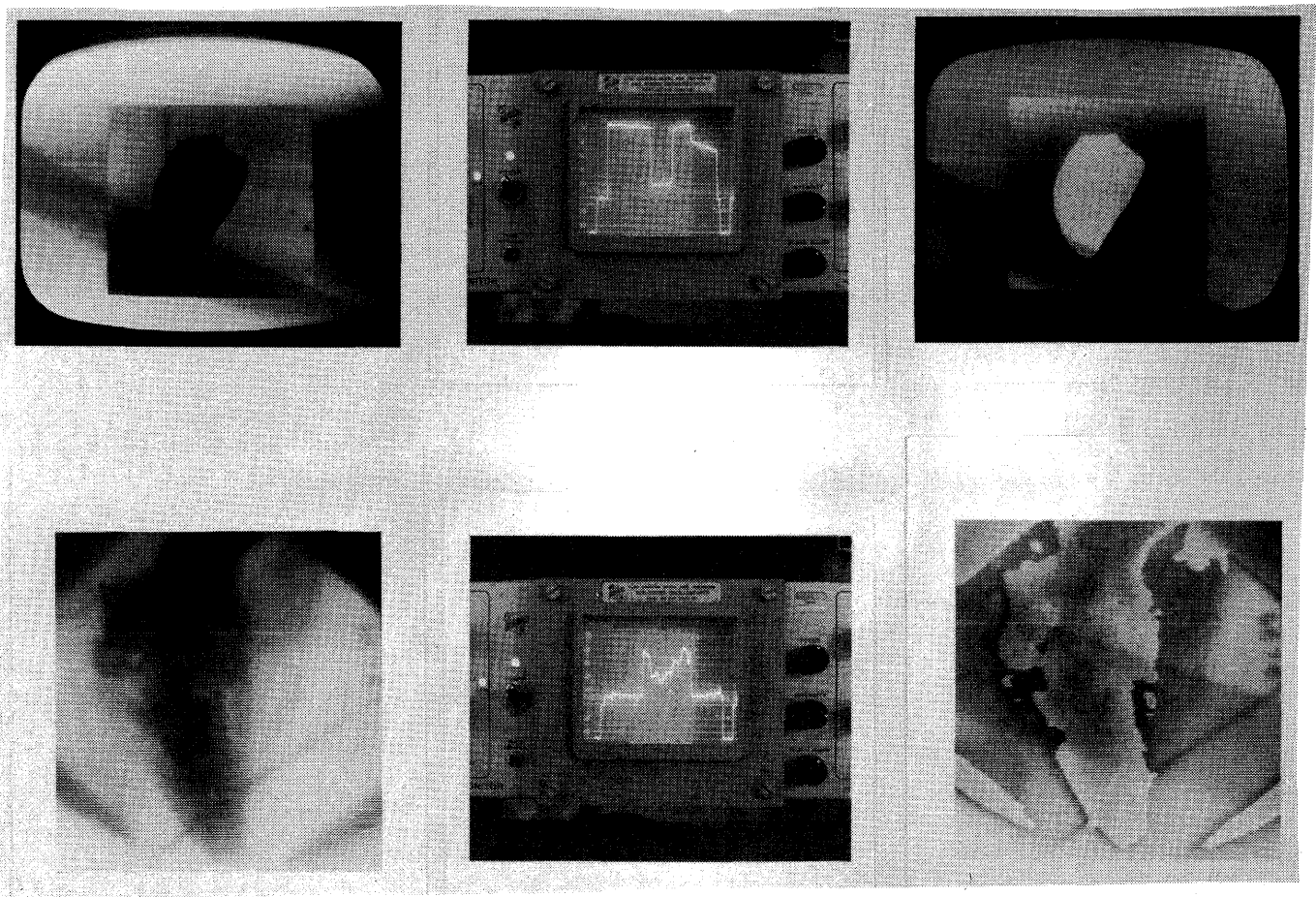


Figure 6—Cutout (top) and film (bottom) television images (left) along with a representative video waveform (center) and the feedback images from the DAI (right). The “window” surrounds both the television and feedback images. The video waveform is from the brightened line that crosses the upper part of the images. The video waveform of the cutouts has sharp edges, high contrast and is uniform, whereas that of the film has a diffuse edge, lower contrast and is not uniform. The result is that the measurement of the cutout is much more accurate and objective than that of film and that small changes in the threshold of the trigger will alter the measured size of the film image but will have little effect on the measured size of the cutout.

The light pen and light pen interface (LPI)

Because obtaining cutouts for use by the dimensional analysis interface is tedious and time consuming, the light pen and light pen interface were developed to input the left ventricular border directly to the computer and to further reduce the time involved in processing dimension data. With the light pen, the image of the left ventricle displayed on a television monitor from 35 mm. film or video tape, is outlined manually, the coordinates of the border being input by the interface directly to the computer. This system is illustrated in detail in Figures 8-10. Two monitors are used in practice, one to allow viewing of the picture with adequate contrast, the other for use as a “tablet”. All the operator needs to do is advance the film frame he

wishes to process, outline the border as he recognizes it by eye, and indicate to the computer that the border is complete. The video disc constantly refreshes the track of the light pen and keys this into the picture the operator is viewing. The computer is interrupted 60 times per second to accept the X and Y coordinate of the position of the light pen. Multiple terminals are thus easily accommodated even by a small computer.

Analog signals: pressure

The trace marker

When first using the dimensional analysis interface (DAI) to process cutouts, pressures were obtained from

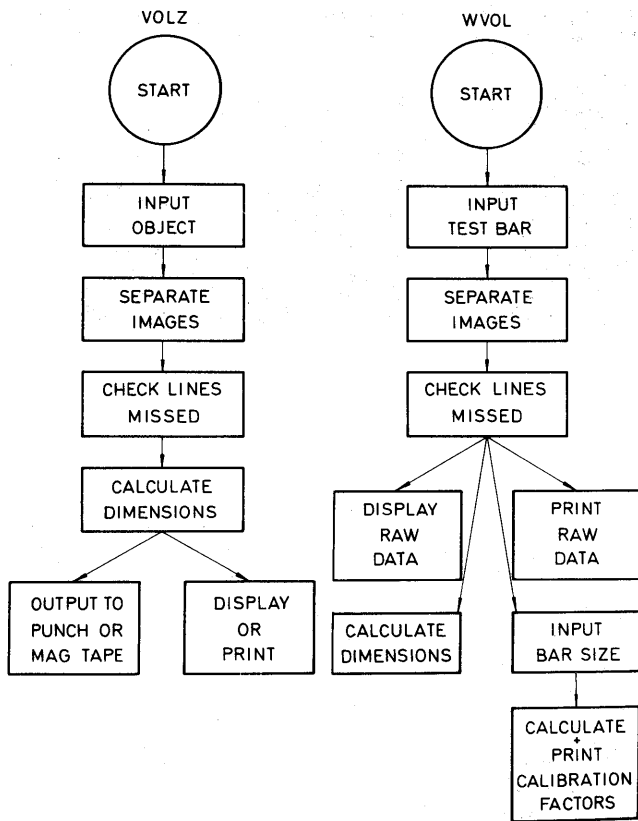


Figure 7—These are flow charts of two programs: VOLZ which calculates the dimensions of images and WVOL which calculates the calibration factors. The latter can be run in a repetitive mode to check the operation of the system and detect electronic faults. Because of the high data rates, both VOLZ and WVOL simply store the measured widths and their corresponding line numbers in assigned memory areas until the buffers are full. Thus, several scans of the same image are available. Checks are done to ensure that no lines were missed and that each discrete image is chosen from the several scans stored in memory. This is done by checking for sequential line numbers.

the oscillographic tracings on which the time of occurrence of each cine frame was marked (Figure 11). A line was drawn by hand from each mark to the left ventricular pressure tracing and the pressures read off, calibrated and tabulated by hand. This method of obtaining instantaneous pressures was used on 80 cases in conjunction with the DAI and cutouts. It is tedious and involves many possible inaccuracies. In particular, there is the possibility of losing the time correlation between the cine frames and the pressure tracing.

To ensure correlation between the film frames and the instantaneous left ventricular pressure it is necessary to guarantee that neither the oscillographic paper nor the film has stopped. Also, frames and trace marker pulses must be counted accurately until the heart cycle of interest is reached. To assure us of the

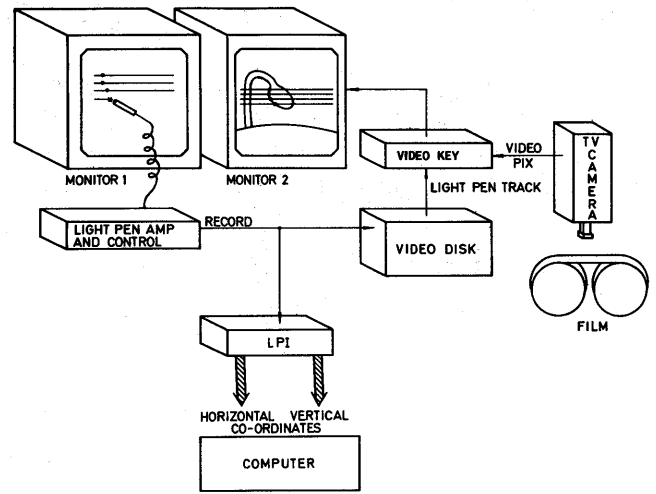


Figure 8—The track of the light pen placed against a monitor displaying a blank, bright raster (monitor #1) can be recorded by the video disc. This track may be keyed (SEG KEY) into the output of a television camera which is viewing a cine film (monitor #2), permitting an operator to outline areas of interest in the picture viewed by the television camera. Simultaneously, the digital horizontal and vertical coordinates of the position of the light pen can be output to the computer by the light pen interface.

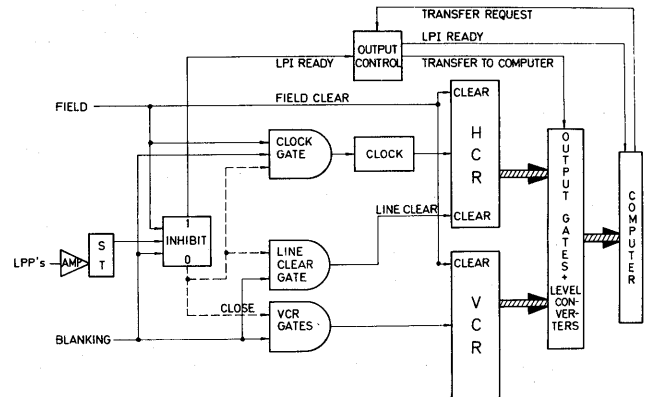


Figure 9—Schematic diagram of the light pen interface (LPI). The FIELD drive pulse clears the HCR (Horizontal Coordinate Register) and the VCR (Vertical Coordinate Register) and provides input conditions to the CLOCK GATE and the INHIBIT. During a television line the BLANKING signal provides a condition to the INHIBIT. At the beginning of each line the BLANKING adds a count to the VCR if the VCR GATE is not inhibited and starts the clock if the CLOCK GATE is not inhibited. The output of the clock is recorded in the HCR. If a light pen pulse (LPP) does not occur before the end of a line, the BLANKING clears the HCR through the LINE CLEAR GATE. If an LPP occurs during a line, a Schmitt trigger (ST) turns the INHIBIT on. The latter INHIBITS the VCR gate, the LINE CLEAR GATE and the CLOCK GATE effectively freezing the contents of the HCR and VCR. The INHIBIT also provides a signal to the output control indicating to the computer that the contents of the HCR and VCR are available for transfer.

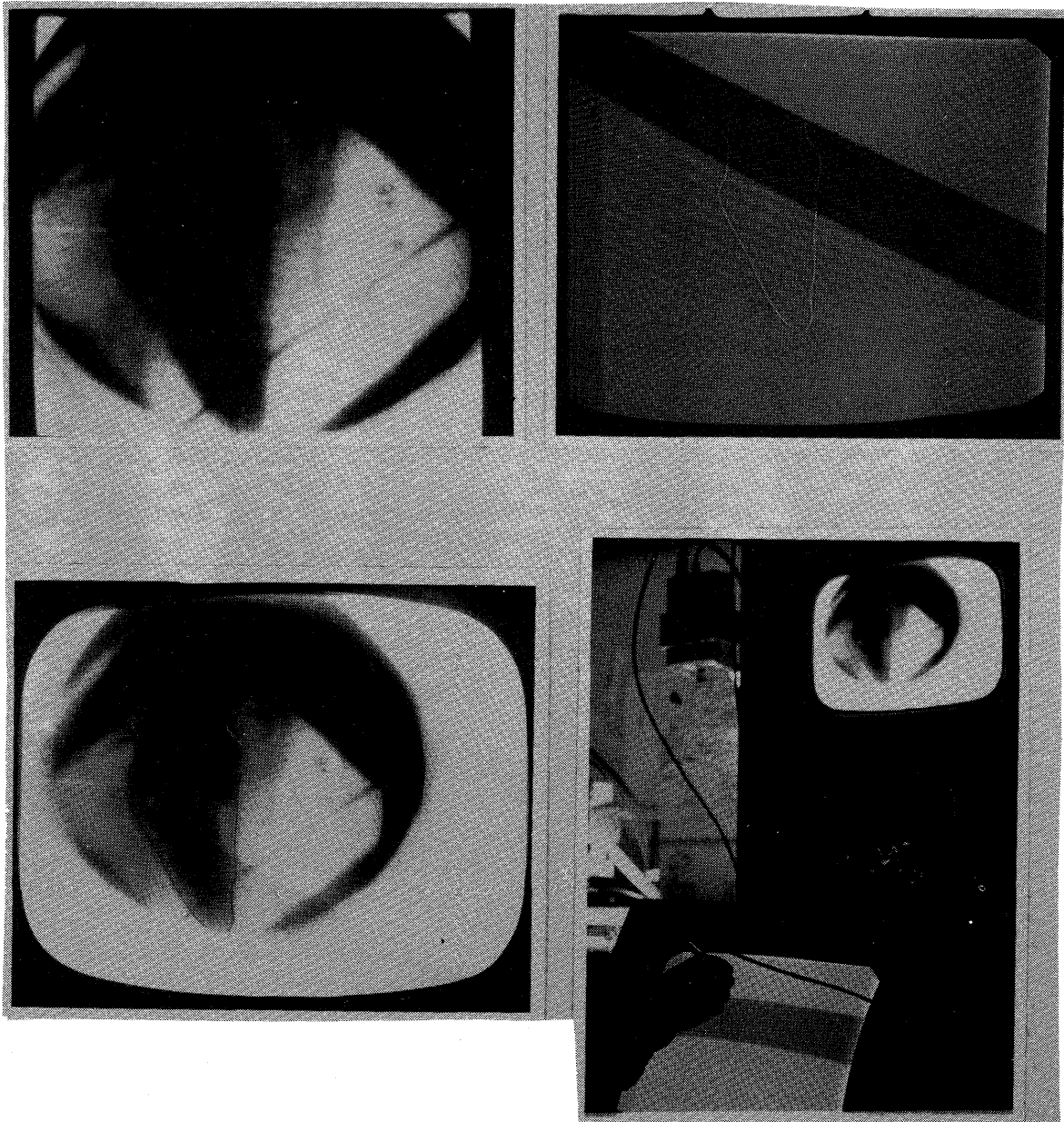


Figure 10—The track of the light pen on a blank, bright raster (top, right). This track has been keyed into a picture of a left ventricular cineangiogram (bottom left) and imaged by a television camera onto a second monitor (top left). By using the bright, blank raster as the drawing tablet and by following the border as it appears on the picture, the operator has been able to outline the left ventricular silhouette (bottom right) with the recorded track.

simultaneity of the two records, a second trace marker was added which has every tenth mark accentuated and which uses this accentuated mark to place a bright dot on every tenth film frame. In this way the number of errors due either to stopped recordings or counting mistakes has been greatly reduced.

The frame marker

Another way of removing the correlation and frame counting problems is to place the pressure directly on the film frame and number the frames. For ease and flexibility this was done by keying the analog pressure

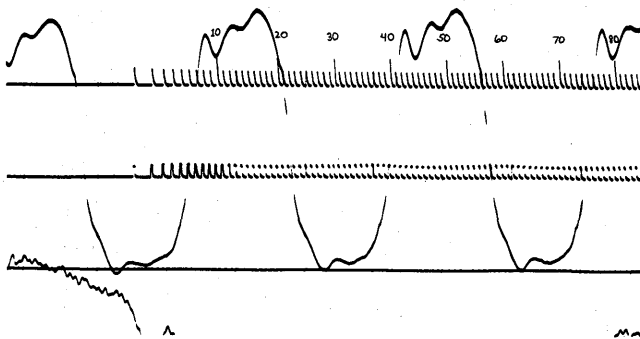


Figure 11—Pressure is recorded using a Statham pressure gauge and displayed on a channel of the Electronics for Medicine oscillographic recorder. Along with the pressure trace there are two markers, one recording the output of a photodiode-fluorescent screen combination in the pulsed X-ray beam and one redisplaying this and accentuating every tenth pulse to facilitate counting up to the particular heart cycle desired. Each pulse corresponds to the exposure of a cine film frame. A mark also appears on every tenth film frame to ensure trace-to-film correlation.

For manual digitization of the pressures a line is drawn using these markers as a reference, and the point at which the line intersects the pressure tracing recorded in units of height. These values are later calibrated to true pressure. The change-over to semi-automatic digitization involves sampling the recorded pressure waveform at the time of occurrence of the peaks on the marker channel.

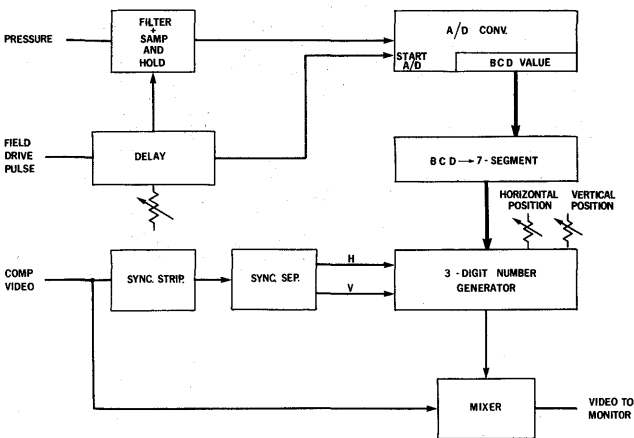


Figure 12—Block diagram of the numerical pressure display on film. The pressure (appearing as a voltage waveform) is filtered to remove high frequency noise, and, at a resetable time after the beginning of each television field, is sampled and analog-to-digital converted to a Binary Coded Decimal number. This number is decoded to 7-segment format. The number generating circuits then create numbers on the screen by keying a bright pattern into the video at the position set by the horizontal and vertical position controls.

signal (displayed as a horizontal bar), the frame number, and the digital value of the pressure into the video output of the X-ray system (Figure 12) and kinescope recording this on 16 mm. film. Alternatively, an optical system is being developed to superimpose these signals directly on the 35 mm. cine film. The pressure bar may be measured automatically by the DAI. The numerical indication of pressure is useful for manual keyboard entry into the computer while using the LPI. In each case, the computer calculates true pressure after being given the appropriate scale factors.

CONCLUSION

It should be noted that this system may be useful in other areas where dimensional or geometrical analysis of pictures is desired, e.g., area measurement of plane objects (DAI), or measurement of the shape and size of chromosomes (LPI). In addition, analog data presented as bars in a television picture or as televised graphs or traces may be handled very easily; in the first case automatically through the DAI or, in the second, manually through the LPI. The LPI may also be used to count and mark objects in a picture.

Lastly, the entire computer system is available as a general purpose laboratory data acquisition and processing system. We have used it for regional myocardial blood flow measurements, for the processing of Xenon washout curves from the lungs, for statistical analysis, for plotting, for digitizing selected television lines and for the analog-to-digital conversion of electrocardiograms.

Using the television/computer system we have processed, in a 3½ month experimental run, the pressures and dimension data from one complete cardiac cycle (60-100 frames) for each of 80 patients. Other routine studies are under way on both normal left ventricles and on pre- and post-operative left ventricular function. The Cardiovascular Unit at Toronto General

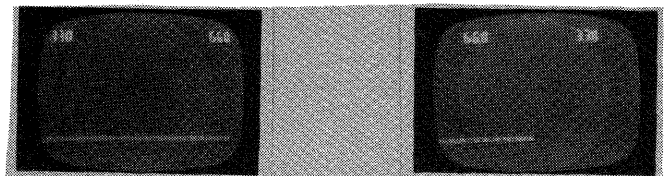


Figure 13—Pictures of a television monitor with the pressure bar and the numerical display mixed with the X-ray video. The number on the left in each picture is the frame number, the one on the right is the numerical value of the pressure. Two different frames have been simulated to show how the display would look for two different pressures.

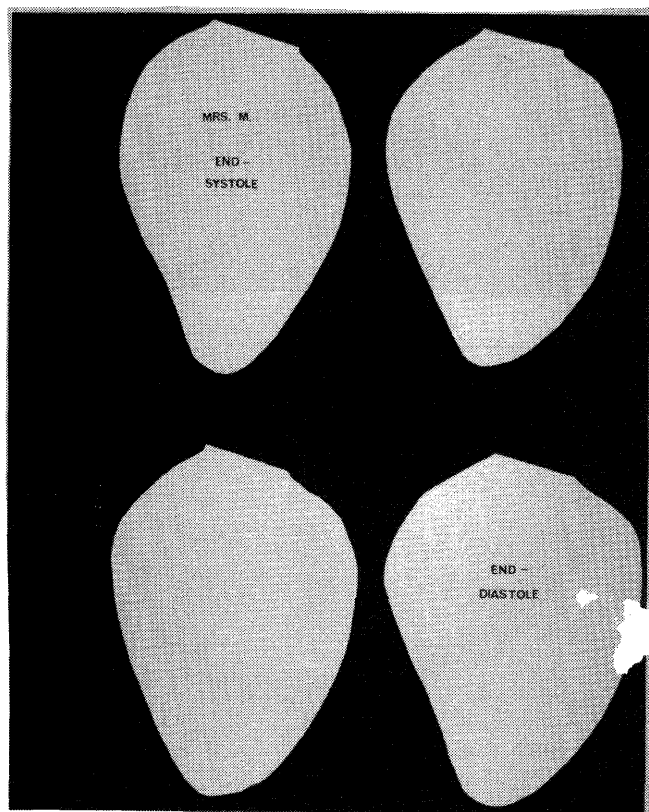


Figure 14A—Mrs. M's ventricle exhibits very poor contraction, a more rounded shape (more spherical than cylindrical) and relatively no change in the long axis length or widths perpendicular to this axis.

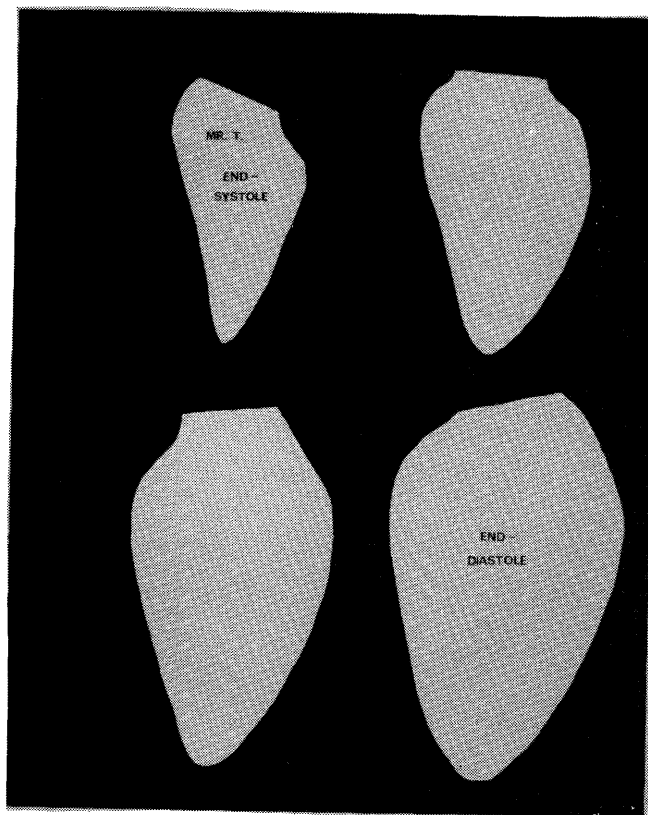


Figure 14B—Mr. T's ventricle has excellent left ventricular contraction, an elongated shape (more cylindrical than spherical) and major changes in the long axis and widths.

Figures 14A and 14B—Geometry. Four frames from a left ventricular cineangiogram are shown, one at the end (end-systole) and one at the start (end-diastole) of the cardiac contraction cycle and two intermediate.

Hospital averages 6 investigations per day. We have thus been able to process roughly 20 percent of the cases available at the Unit. Using the hand techniques for measuring volume, we would have been able to process (obtaining volumes only) less than 5 percent of the available cases. It is projected that we will be able to process 50-75 percent of the available cases with the total implementation of the light pen interface and the numerical pressure indication system. This increase can be achieved without the addition of any staff, and with no additional expenditures on hardware. Furthermore, the system will be capable of providing more advanced information to the clinical staff than is currently available.

RESULTS

To illustrate the kind of information generated by the system the results of studies done on 2 patients,

one with very poor left ventricular function and one with excellent left ventricular function, are shown in Figures 14-19. Figure 14 shows that the poorly contracting ventricle exhibits little change in all dimensions (width, length and area), whereas the normal ventricle, shows marked reduction in its dimensions. Figure 15 illustrates that the left ventricular widths are larger and show little change from the start to the end of contraction in the poor ventricle compared to the normally contracting ventricle. Similarly, in Figure 16, the poor ventricle's volumes are larger, the rise time slower, and variation smaller in contrast to the normal left ventricle. The pressures (Figure 17) show a slower rise time and a higher minimum pressure in the poor ventricle compared to the good one. The pressure-volume correlation shown in Figure 18 demonstrates that the stroke work (i.e., the work done in ejecting blood) is much smaller in the poorly contracting ventricle, that this work is done at a higher pressure in a

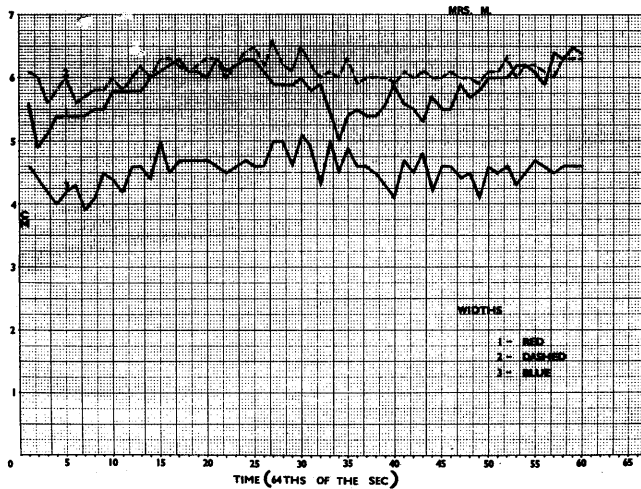


Figure 15A—It should be noted in Mrs. M's width plots, that the values are large, that there is little change from end-systole (small widths) to end-diastole (large widths), and that the rise times are slow. The width at the base of the heart (the top, width 1) shows relatively better contraction compared to the other two widths. Width 3 (at the apex) is out of phase with number 1. These findings are in keeping with the visual findings of a lack of contraction in this area.

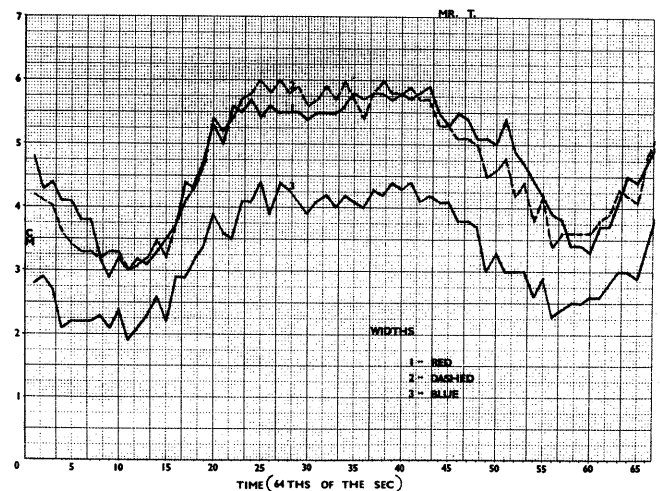


Figure 15B—In Mr. T's width plots, the marked change in widths is apparent. The rise times are fast, indicating vigorous contraction. All the traces are roughly in phase, indicating that all areas are contracting together.

Figures 15A and 15B—Width plots. The plot of three widths from the left ventricle at $\frac{1}{4}$, $\frac{1}{2}$, and $\frac{3}{4}$ of the way along the long axis.

larger chamber, and that little blood is ejected. This means that the poor ventricle has a mechanical disadvantage relative to the normal ventricle. Finally, the wall tension (Figure 19) starts higher, rises more slowly, and remains higher longer in the poor ventricle compared to the good one, indicating that the poor ventricle uses more energy to do less stroke work.

Accuracy in these studies

The accuracy of the measurements of the left ventricle is most affected by picture quality, e.g., if the border of the left ventricle is blurred, the position of the edge cannot be precisely determined. Various arbitrary criteria may then be used to determine the edge automatically and each will produce different measurements of the size of the left ventricle. For this reason, we have chosen to rely on the eye to choose the border. The eye is superior to the machine in recognizing the boundary since it is very sensitive to small changes in contrast and since it uses the whole picture to assist in

judging the border. This is put to use in the LPI and in the making of cutouts or the selection of the threshold when using the DAI.

The DAI, when measuring films and cutouts from films of model ventricles, was generally within five percent of the true volume of the model.²⁵ In routine X-ray films, however, the measured size of the ventricle is very dependent on the threshold chosen. In these cases, the eye must be used to judge the position of the border.

But, even the eye may find it difficult to choose a border, since the border may be very unsharp or irregular due to poor mixing of the radio-opaque dye and the non-uniformities of the wall itself. Some of these problems may be resolved by better injections, improved picture quality, more understanding of the attenuation of the X-rays by the contrasted heart, and more knowledge of the behavior of the left ventricular wall irregularities (trabeculae carneae and papillary muscles) during the heart cycle. Work needs to be done in all of these areas to improve the accuracy and meaning of the results obtained. This is particularly

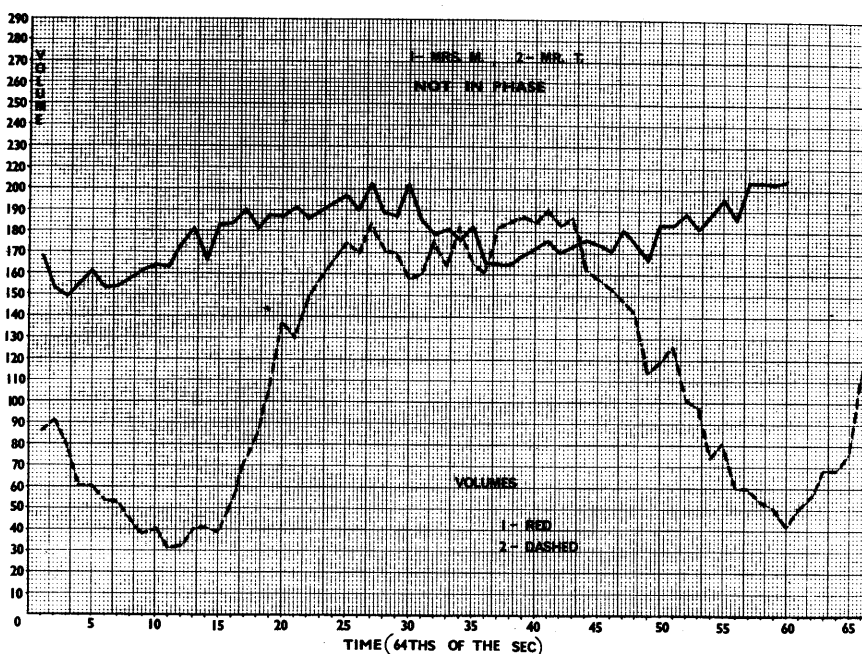


Figure 16—The volume curves. It should be noted that these curves are displayed on one sheet for convenience only and that they are not in phase, since the two hearts are beating at different rates.

Mrs. M's (#1) ventricular volumes are large and there is little change in volume during contraction. The rate of change of volume is slow and the usual characteristic features of volume changes during a heart cycle are not evident.

In Mr. T's (#2) curve there is a large change in volume with a good rise time. Although noisy, the curve seems to show a rapid-filling phase (frames 15-20), a reduced-filling phase (frames 20-27), atrial systole (the contribution of the contraction of the atrium) (frames 36-43) and an ejection phase (frames 43-60). These features are not apparent in Mrs. M's ventricular volume plot.

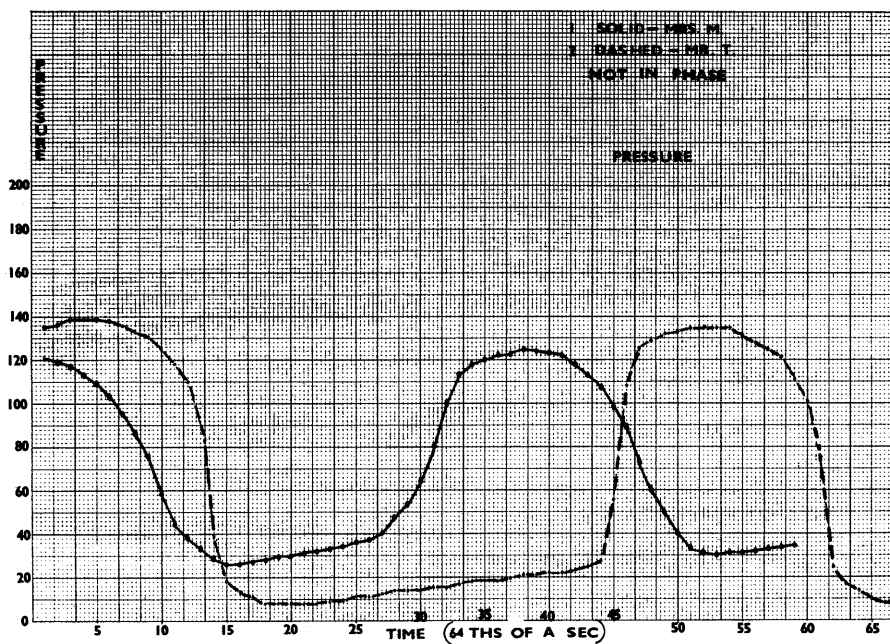


Figure 17—Pressure plots. These are the pressure tracings for the two ventricles. The curves are not in phase. The features that are important are: Mrs. M's peak pressure is lower than that of Mr. T, whereas Mr. T's minimum pressure is lower than that of Mrs. M. The higher minimum (diastolic) pressure in Mrs. M's ventricle is typical of a large, dilated, poorly contracting ventricle. Mrs. M's pressure rise is more gradual, indicating a more poorly contracting ventricle than Mr. T.

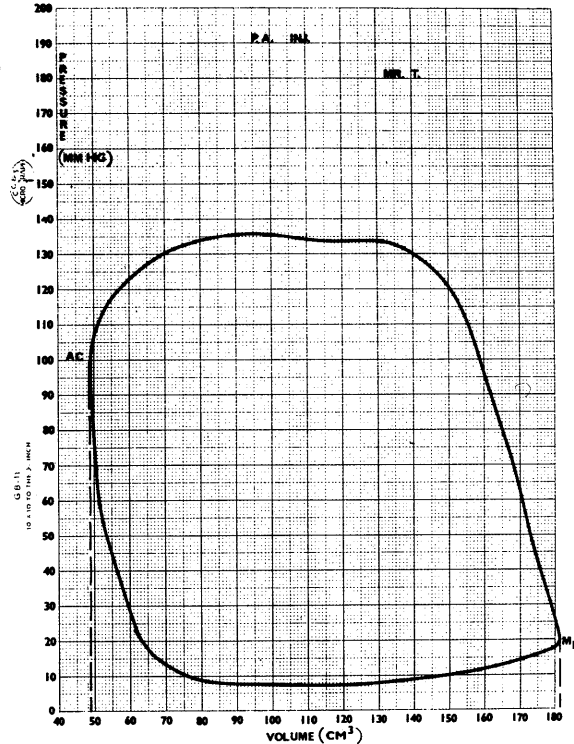
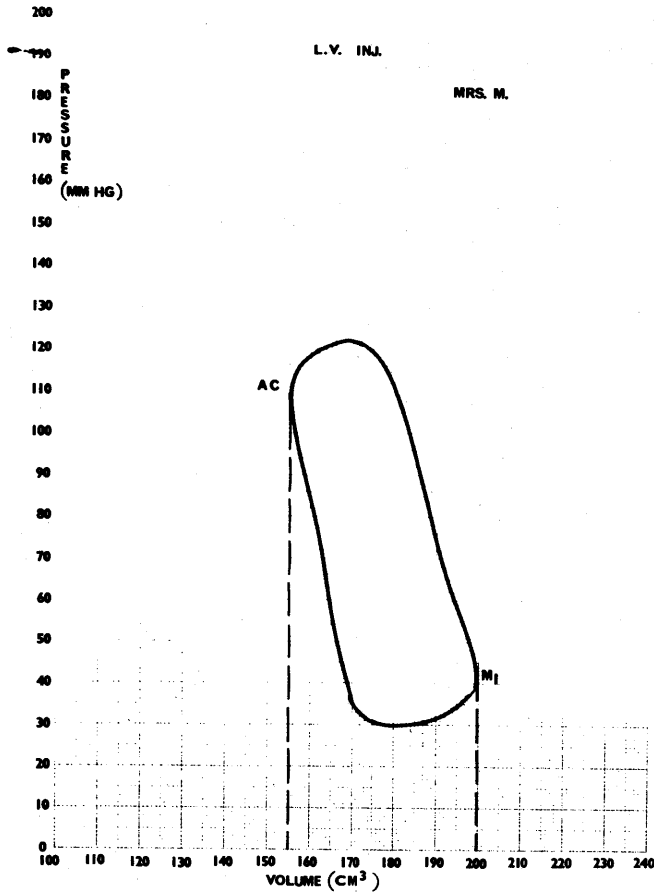


Figure 18A—The main features of Mrs. M's loop are that its area, which is proportional to the stroke work, is small. The volume of blood ejected relative to the volume of the cavity is small. Moreover, the centroid of the loop is shifted upward (high pressure) and to the right (high volume) compared to a normal ventricle.

Figure 18B—Mr. T's plot is characterized by a high stroke work (area enclosed by the loop) and a shift in the centroid of the loop downward and to the left as compared to Mrs. M.

Figures 18A and 18B—Pressure-volume plots. Smoothed pressure-volume plots for Mrs. M and Mr. T.

true now, as we would like to measure wall thickness accurately.

Other errors in left ventricular measurements arise from changes in the X-ray system magnification (image intensifier), spatial distortion by the intensifier, film shrinkage, human error in outlining and cutting out the paper for the cutouts, the spatial orientation of the left ventricle when it is filmed, and the orientation of the cutout while being measured. Studies of these errors are in progress and techniques will be refined to reduce them.

FUTURE OBJECTIVES

The light pen and light pen interface are becoming operational on a routine basis. This speeds the processing time, allows more cycles per patient to be done, and

provides more information per frame (the coordinates of the border are available and hence shape may be measured). With the ability to process a larger number of heart cycles per patient, we should be able to study the stressing effects of drugs and other inotropic agents, such as pacing, within a given angiogram or by repeat angiograms. This should provide further information about the functional state of the left ventricle.

Meanwhile, a systematic study of the X-ray system and other factors affecting border recognition is being pursued. This should eventually provide us with better quality pictures.

A number of projects are under way in the development of the television/computer system:

- (1) We will attempt to improve the automated system (DAI) by using more reliable border recog-

dition criteria (presently merely voltage threshold sensed by a Schmitt Trigger) and also by enhancing the image presented to it, finally perhaps carrying out rapid measurements on videotape recordings and displaying the results during the catheterization.

- (2) We are planning a more efficient semi-automated system involving improvements to the LPI and the use of the DAI in conjunction with it; for example, the LPI can be used to delineate areas to be quickly and automatically measured by the DAI (the left atrium and aorta may be cut off using the LPI, and the left ventricle measured by the DAI).
- (3) More use of automation in the handling of analog data is being attempted.
- (4) We are considering the development of a videodensitometric system to provide two important measurements: (a) of the blood flow, and (b)



Figure 19—Wall tension plots. These plots of the wall tension of each ventricle are not in phase. The tension plots show Mrs. M starting at a higher tension than Mr. T, rising more slowly, peaking higher and later, and remaining higher longer than Mr. T. Mrs. M's curve is characteristic of a dilated poorly functioning left ventricle, and Mr. T's of a fairly normal ventricle.

of the depth of opacified objects (assuming that the density of an image is due to the depth of absorbing material in the path of the X-rays).

Presently, the system is being assessed as to its usefulness in the clinical and biophysical studies now being carried out.²⁷ It has, however, already been valuable in providing clinical data in the Cardiovascular Unit and may have many applications for measurements on pictures and analog signals elsewhere within the hospital, in research, and in industry.

ACKNOWLEDGMENTS

The authors are grateful to the technical staff of the Department of Medical Engineering and Biophysics, particularly to Mr. Paul Mendler for his work on the DAI and LPI, to Mr. Roy Liggins for his excellent technical assistance, to Mr. Donald Mills for constructing the DAI, to Mr. Robert Kubay for building the trace marker circuits, to Mr. Robert Growcock for the digital number display construction, and to Mr. Franz Schuh and Mr. Louis Rostocker for their mechanical assistance. In addition, the authors would like to express their appreciation to Mr. Eric Covington for his contribution to the programming, to Mrs. Yasna Polic, Mrs. Ulla Nordin and the Department of Art as Applied to Medicine for the diagrams, to Mr. Barry Bassett for the photographs in Figures 6, 10 and 13, to the Department of Medical Photography, Toronto General Hospital for photographing the figures, and to Miss Vivian Martin for her excellent secretarial assistance. This work was supported by the Ontario Heart Foundation and Toronto General Hospital.

REFERENCES

- 1 A BURTON
The importance of the size and shape of the heart
Vol 54 No 6 p 801 1957
- 2 H T DODGE W A BAXLEY
Left ventricular volume and mass and their significance in heart disease
The American Journal of Cardiology Vol 23 p 528
April 1969
- 3 E H SONNENBLICK W W PARMLEY
C W URSCHEL
The contractile state of the heart as expressed by the force-velocity relations
The American Journal of Cardiology Vol 23 p 488
April 1969
- 4 I L BUNNELL C GRANT D C GREENE
Left ventricular function derived from the pressure-volume diagram
American Journal of Cardiology Vol 39 p 881 December 1965

- 5 J H GAULT J ROSS E BRAUNWALD
Contractile state of the left ventricle in man
Circulation Research Vol XXII p 451 April 1968
- 6 H L FALSETTI R E MATES C GRANT
D C GREENE I L BUNNELL
Left ventricular wall stress calculated from one-plane cineangiography
Circulation Research Vol XXVI p 71 January 1970
- 7 A Y K WONG P M RAUTAHARJU
Stress distribution within the left ventricular wall approximated as a thick ellipsoidal shell
American Heart Journal Vol 75 No 5 p 649 May 1968
- 8 D D STREETER R N VAISHNAV D J PATEL
H M SPOTNITZ J ROSS E H SONNENBLICK
Stress distribution in the canine left ventricle during diastole and systole
Biophysical Journal Vol 10 p 345 1970
- 9 R NATHAN
Digital video data handling
Technical Report No 32-877 Jet Propulsion Laboratory
California Institute of Technology January 5 1966
- 10 D A WINTER B G TRENHOLME D MYMIN
E L MYMIN
Computer processing of videoangiographic images noise reduction image enhancement and data extraction
Canadian Cardiovascular Society Conference October 15-17 1970
- 11 C K CHOW T KANEKO
Boundary detection of radiographic images by a threshold method
Unpublished manuscript received from the IBM Corporation Thomas J Watson Research Center
December 1970
- 12 R J GOERKE E CARLSSON
Calculation of right and left cardiac ventricular volumes
Investigative Radiology Vol 2 p 360 September-October 1967
- 13 M E SANMARCO S H BARTLE
Measurement of left ventricular volume in the canine heart by biplane angiocardiology: accuracy of the method using different model analogies
Circulation Research Vol XIV p 11 1966
- 14 A G TSAKIRIS D E DONALD R E STURM
E H WOOD
Volume ejection fraction and internal dimensions of left ventricle determined by biplane videometry
Federation Proceedings Vol 28 No 4 p 1358 1969
- 15 P H HEINTZEN V MALERCZYCK
J PILARCZYCK K W SCHEEL
A new method for the determination of left ventricular volume by use of automatic video data processing
Abstract American Heart Association 43rd Scientific Sessions 24th Annual Meeting 1970
- 16 M R GARDNER H R WARNER
Dynamic aortic diameter measurement in vivo
Computers and Biomedical Research 1 p 50 1967
- 17 C B CHAPMAN O BAKER J H MITCHELL
R G COLLIER
Experiences with a cinefluorographic method for measuring ventricular volume
The American Journal of Cardiology 18 p 25 1966
- 18 A JARLOV T MYGIND CRISTIANSSEN
Left ventricular volume and cardiac output of the canine heart
Medical and Biological Engineering Vol 8 No 3 p 221 1970
- 19 A H GOTT
Cooperative heart study cardiac volume analysis
SPIE Journal Vol 8 p 233 1970
- 20 H T DODGE SANDLER D W BALLEW
J D LORD JR
The use of biplane angiocardiology for the measurement of left ventricular volume in man
American Heart Journal 60 p 762 1960
- 21 D DAVENPORT G J CULLER J O B GREAVER
R B FORVAREL W G HANEL
The investigation of the behaviour of microorganisms by computerized television
IEEE Transactions on Biomedical Engineering Vol BME-17 No 3 p 230 1970
- 22 M L MARCUS W SCHUETTE W WHITEHOUSE
J BAILEY D L GLANCY S E EPSTEIN
A completely automated video tracing technique for the determination of dynamic changes in ventricular volume
Abstract American Heart Association 43rd Scientific Sessions 24th Annual Meeting 1970
- 23 H D COVVEY
Measuring the human heart with a real time computing system
Data Processing Magazine p 27 May 1970
- 24 H D COVVEY A G ADELMAN
C H FELDERHOF E D WIGLE K W TAYLOR
The television/computer dimensional analysis interface
Submitted for publication March 1971
- 25 C H FELDERHOF A G ADELMAN
H D COVVEY E D WIGLE K W TAYLOR
Unpublished Data
- 26 H D COVVEY
The measurement of left ventricular volumes from T V with a PDP-8/I
The proceedings of the Digital Equipment Computer Users Society Atlantic City New Jersey p 255 Spring 1970
- 27 H D COVVEY
A television/computer system for the rapid processing of x-ray pictures and analog signals in studies of the left ventricle
Master of Science Thesis University of Toronto
January 1971

Cost benefits analysis in the design and evaluation of information systems

by I. LEARMAN

Medical Systems Technical Services Inc.
Rolling Hills, California

INTRODUCTION

In June of 1969, a report¹ was prepared for the Federal Hospital Council by the staff of the Health Care Technology Program of the National Center for Health Services Research and Development. The report entitled *Summary Report on Hospital Information Systems*, has as primary objectives—"to give a broad view of the components of automated information systems, to briefly evaluate the cost and effectiveness of such systems, and to estimate their future importance."

In meeting the first and last objectives, the report is quite good and should be read by all who are interested in the field. It is in the areas of cost and effectiveness that the report is weak. This in no way should be considered a reflection upon the authors, who fully recognize the paucity of data in these matters. In fact, the authors wisely address, first the moneys being spent on hospital information systems, and then independently, their general performance and acceptance. Indeed, among the quite excellently thought out conclusions is the following:

"The discrepancy which exists between the apparent success and enthusiasm for the use of the computer (sic) in the business and chemistry applications as opposed to the patient management areas suggest that the need for and utility of the computer were more easily recognized, which resulted in a high degree of motivation to see these projects through to a successful operational stage. It might then be assumed that the lack of success in other areas has been a result of an inability on the part of our hospitals to precisely define either the need or practical utility that the computer can serve in other patient management areas. We would be willing to speculate that until such time as other medical services, independent of external pressures, are capable of first recognizing and then demanding more efficient utili-

zation of their time and services, attempts to automate these activities will continue to fail."

It is the author's judgment that, concomitant with the increased recognition and demand of other medical specialties for computer applications is the need for reliable and acceptable methods of assessing the practicality and effectiveness of those applications. The paper attempts to address such methods of assessment. Specifically, we will discuss absolute cost effectiveness—the measure of worth of applying technology to the enhanced transfer and processing of information. The segments of technology include analysis, design, implementation, operation and maintenance. These are measured in the utilization of available resources employed to each technology segment—people, equipment, material and facilities.

The paper will not address relative cost effectiveness—the optimization of resource selection. For such discussions, the reader is referred to a paper² on the subject published by the author. In what follows, a case will be presented for the establishment of absolute cost effectiveness parameters during the planning and design of a hospital information system.

THE ANALYSIS OF COST EFFECTIVENESS IN DESIGN

The steps in cost effectiveness analyses take the following general sequence:

- Selection of Domains of Analysis
- Analysis of "Current" Operations
- Determination of Absolute Cost Effectiveness
- Relative Cost Effectiveness
- Refinement of Absolute Cost Effectiveness

Note the relative cost effectiveness (optimization of

MAJOR AREAS		
MANAGEMENT SYSTEMS	HOSPITAL	INSTRUCTIONAL SYSTEMS
Student Records	Nursing Units	Computer Aided Instruction
Resource Allocation	Surgery	Computer Managed Instruction
Personnel	Pharmacy	Availability of Instructional Data
Payroll Administration	Radiology Clinical Laboratories	
Accounts Receivable	Medical and Dental Records	
Accounts Payable	Dietary	
Purchasing	Hospital Business Office Insurance Office Admitting and Patient Logistics	

Figure 1—Cost analysis, goal-hard reductions in required resources

resources) is considered in the context of this paper as a refinement of the measure of worth. In the evaluation of cost effectiveness, this step would not be available.

Domain of analysis

In order to attain a valid measure of worth, it is essential to compare old apples to new apples or manual oranges to automated oranges. We call the sum of all

PROCESS	ADMITTING CLERK	
	PRESENT	1974
Reservations	3	3.6
Bed Availability Check	.33	.33
Nursing Unit Check	.33	.33
Bed Control Card	.33	.33
Type Daily Admissions List	.5	.6
Type Transfer List	.15	.18
Notify Surgery and Nursing Units of Room Change	.15	.15
Check Discharges, Pull and Mark for Information Desk	1	1.2
Emergency Admissions	.25	.3
Assign Bed and Notify Units	1	1.2
Prepare Pre-Admit Form	.5	.6
Handle Patient Transfer Requests	1	1.2
Call for Pre-Admit Information	10	12
Search for Pre-Admit Information	5	6
Type Admission Form	10	12
TOTALS	33.55	40.02
NUMBER OF PERSONNEL	4	5

Figure 2—Result of cost analysis, current costs admissions and reservations information handling, current system in hours per day

valid areas of measurement the domain of analysis. The criteria for selecting valid areas for current (manual, automated, or modeled) and new (improved manual or automated) information handling applications are as follows:

First, each application must have a common, definable entry and output. Second, the designed change must have some implication to benefits. Third, the application in the current system must have a functional equivalence relationship to the new system.

Some examples may be in order:

Application: Laboratory Order to the Lab

Criterion 1—Order entered at ward 1, order output at Lab

Criterion 2—Faster turnaround time, reduce transcription, error.

Criterion 3—The current system requires the physician to write the order, the nurse to transcribe it, the ward clerk to send it through the tube, the secretary to log it, separate it, and send it to the proper department(s).

The new system may require the physician to enter the order directly into a device. The nurse and the proper laboratory department(s) receive it via output devices. But the function was equivalent. This would identify a valid area for our domain of analysis.

PROCESS	ADMITTING CLERK
	1974
Reservations	1
Bed Check	.1
Nursing Unit Check	0
Bed Control Card	0
Type Daily Admission List	0
Transfer List	0
Notify of Room Change	0
Check Discharges for Information Desk	0
Emergency Admissions	.1
Assign Bed	.33
Prepare Pre-Admit Form	.5
Handle Patient Transfer Requests	.5
Call for Pre-Admit Information	5
Search for Pre-Admit Information	.5
Enter Admission Data	10
Set Up Admit Package	3
Notify of Patient Arrival	.33
Indicate if Admit Lab.	.33
TOTAL	21.69
NUMBER OF PEOPLE	3

Figure 3—Result of cost analysis—New system, admissions and patient logistics information system, requirements in hours per day

CURRENT SYSTEM										
	1970	1971	1972	1973	1974	1975	1976	1977	1978	1979
Admissions Clerks	4	4	4	5	5	5	5	6	6	6
ADMISSIONS AND PATIENT LOGISTICS INFORMATION SYSTEM										
Admissions Clerks	4	3	3	3	3	3	3	3	4	4
DIFFERENCE TABLES										
Admissions Clerks	0	1	1	2	2	2	2	3	2	2
Annual Reductions	0	\$7,000	\$7,350	\$15,000	\$15,700	\$16,500	\$17,300	\$26,000	\$18,500	\$19,500
Monthly Reductions	0	585	615	1,250	1,310	1,375	1,650	2,330	1,560	1,625
5 Year Annual Average	\$9,010									
5 Year Monthly Average	750									
10 Year Annual Average	14,280									
10 Year Monthly Average	1,175									

Figure 4—Results of cost analysis, admissions and reservations cost comparisons

Figure 1 lists the major functional areas in a teaching hospital where significant numbers of such areas of analysis have been found.

Cost analysis—Quantitative benefits

Hard savings

For each area in our domain of analysis, a cost analysis is performed. The sequence is as follows:

- Selection of Applicable Procedures
- Determination of "Current" Resource Requirements
- Statement of Growth Assumptions
- Extrapolation of Resource Requirements to Operational Era
- Determination of "System" Resource Requirements
- Comparison of Extrapolated "Current" to "System"

Each procedure in the area is defined together with the current required resources to accomplish the procedure over an operational duration (minute, hour, shift, day, week, month). An extrapolation of required hospital resources is then determined for the same operational time frame as the new system (2 years hence, 5 years hence, etc.).

The new information system design will point to a different set of resources to accomplish the tasks. During this phase of the analysis, no attempt is made to include development and operational costs for the information system; only those differences in carrying out the tasks in the domain are compared.

The cost entities included in these analyses are personnel salaries, overhead and fringe benefits; disposable material such as forms or cards; equipment capital costs, rentals, and maintenance; facilities requirements

for offices, storage, equipment, etc. These cost entities must be established for both the extrapolated "current" operations as well as the "system" operations.

Figures 2, 3 and 4 demonstrate the results of such an analysis for admitting clerks.

Figure 2 demonstrates the procedures identified for analysis and the present and extrapolated labor.

Figure 3 demonstrates the same procedures but with the labor resources required in the new system. Figure 4 compares the two and summarizes the potential hard savings.

We use the term "hard" to describe cost savings which can be taken to the bank—reduction in personnel, material, equipment. In Figure 5, the column labeled "Reduction Assumed" (R) includes all such savings in costs per full operational month.

Partial time savings

The most common source of error found in reviewing cost effectiveness analyses, has been in the quantification of part-time labor savings. It is tempting to sum all the minutes and hours of partial time saved and to include the total in the quantitative benefits. The error

Area	Total Information Processing	Total Potential Savings	Reductions Assumed (R)	Partial Time Saved (P)
Business Office	\$ 20,050	\$ 11,785	\$ 8,300	\$ 3,485
Admitting	3,650	2,000	1,310	690
Pharmacy	8,110	5,960	5,500	460
Laboratory	25,142	12,315	9,700	2,615
Nursing Units	127,400	85,400	15,000	70,400
Surgery	2,990	1,500	1,250	250
Radiology	3,342	1,928	1,040	888
Dietary	2,150	1,575	1,200	375
TOTALS	\$192,834	\$122,463	\$43,300	\$79,163

Figure 5—Summary of cost analyses for selected major hospital entities, full operating system dollars per month

in doing this is that it implies perfect management—certainly an ideal but never a reality. For it would mean that all of the partial times could be used fully in performing other functions resulting in equivalent benefits. We have found that an administrative efficiency factor should be used to weight such potential benefits properly.

Figure 5 shows the relationship among four quantifiable factors for major areas of a 400 bed hospital.

“Total Information Processing” are those labor dollars spent in the current system within the domains of analysis.

“Total Potential Savings” are all the hours of labor dollars saved by the new system.

“Reductions Assumed” are the savings which we have called quantitative benefits (R).

“Potential Time Saved (P)” are the partial hours of labor dollar savings.

It is the last figure (P) which we will penalize by assigning a $33\frac{1}{3}$ percent Administrative Efficiency Factor. In other words, for every hour of partial time saved, 20 minutes are utilized effectively. Note from the figure that the greatest P exists at the Nursing Units.

Intangible benefits

Perhaps the most difficult role of the analyst is the assignment of values to benefits which are not explicitly measured in terms of identifiable resources. One can simplify the problem monumentally by dividing such benefits into two general types:

- **Plausible Quantitative Measures**—These can be associated with a product of the information flow which is in itself measurable. Examples of these are benefits which can be measured as factors of revenues or costs (improved bed utilization resulting from more timely and effective bed reservation and surgery scheduling systems).
- **Judgmental Criteria**—These are the most difficult to measure and are used as supportive (or deciding) arguments when all measurable data have produced a borderline or negative cost effectiveness picture. Examples include enhanced care of high risk patients, availability of processing power for research support, etc.

Several examples of assigning quantitative values to intangible benefits are described below

- (1) If increased administrative effectiveness can be measured as a percentage of management time,

that factor of administrative payroll can be taken as a benefit.

- (2) Increased bed utilization enhances the effectiveness of patient management. The resulting higher census and better scheduling can be measured as an increase in hospital revenues.
- (3) The transition to, and the operating environment of, a new system can profoundly effect the rate of personnel turnover. The increase or decrease of such turnover can be measured in terms of personnel acquisition costs.

As implied in the last example, such measurements can produce negative results. The Quantitative Benefit (Q) is the sum of these measurements ($\sum E_i X_i$) and is given one-sixth the weight of the hard benefits.

An equation for cost effectiveness (E/C)

The analytic expression

E/C is a function of time based upon the accrual of benefits and costs and can be expressed analytically as follows:

$$E/C = \int_0^L \frac{B(t)dt}{C(t)} \quad (1)$$

where $B(t)$ are the accrual of benefits, $C(t)$ are the accrual of costs, and L is the life of the system including development and implementation.

The author has used two approximations to this equation in performing cost effectiveness analyses.

Approximation #1

$$\begin{aligned} E/C &= (C_{av})^{-1} \int_0^L B(t)dt \\ &= (C_{av})^{-1} \sum_L B(t) \approx \frac{\sum p(B_{max})}{L(C_{av})} \end{aligned} \quad (2)$$

where

$$B_{max} = R_{max} + \frac{P_{max}}{3} + \frac{Q_{max}}{6} \quad (3)$$

The benefits and costs per month accrue over the years of development, implementation, and operation. Using B_{max} as the fully accrued monthly benefits and C_{av} as the average monthly system costs over L years, one can approximate the integral such that

$$E/C = \frac{\sum p(B_{max})}{L(C_{av})} \quad \text{where } \sum p \quad (4)$$

equals the percentages per year of benefits accrued.

In the example which follows, we have used an L of 7 years and a $\sum p$ of 4.5 (made up of .1, .2, .4, .8, 1, 1, 1).

If

$$E/C + \frac{\sum p(B_{\max})}{L(C_{av})} \quad (5)$$

is greater than 1, the system is judged as cost effective. Examining the end points of this equation is an interesting exercise. Should there be no partial time saved (P) or qualitative benefits (Q), then the hard benefits (R) must exceed the total cost of the new system (C_{av}) over the system lifetime span. C_{av} includes all the costs of all the resources as indicated under Hard Savings in the discussion of Cost Analysis—Quantitative Benefits. Now let us assume no hard benefits and no qualitative benefits. Then the total partial labor saved would have to exceed three times the C_{av} . In our example from Figure 5, P equaled approximately \$80,000. The average monthly cost of the major system was \$40,000. Without the other two factors exceeding \$35,510, the system would not have been judged as cost effective since—

$$3 C_{av} = \$120,000$$

$$P = \$80,000$$

$$\frac{4.5 \left(\frac{80,000}{120,000} \right)}{7} = \frac{9}{21} = .429$$

$$\frac{4.5(R + Q/6)}{7(40,000)} > .571$$

$$R + Q/6 > \$35,510$$

Then $E/C > 1$

Although unlikely, the other extreme case would mean there were only qualitative benefits. Using the same example, Q would have to exceed \$360,000 if R and P were zero, in order for E/C to be greater than 1.

The actual example turned out the following results:

$$\begin{aligned} E/C &= \frac{4.5}{7(40,000)} \left(43,000 + \frac{80,000}{3} + \frac{185,000}{6} \right) \\ &= \frac{4.5}{280,000} (108,000) \\ &= \frac{485,000}{280,000} \\ &= 1.73 \end{aligned}$$

The following table indicates the cost effectiveness

values for our example under varying extremes:

	E/C
$R = 0$	1.04
$R = 0, P = 0$.64
$R = 0, Q = 0$.43
$P = 0$	1.3
$P = 0, Q = 0$.69
$Q = 0$	1.12

These figures indicate that for our example, cost effectiveness could not be achieved by considering one benefit category only and cost effectiveness could be achieved considering any two benefit categories. Approximation #1 was found to be most useful in determining cost effectiveness of integrated hospital information systems.

Approximation #2

$$E/C_L = \frac{R_L}{C_L} + \frac{P_L}{C_L} + \frac{\sum^L p(Q_L \max)}{LC_L} \quad (6)$$

where R_L and P_L are the cumulative benefits up to time L , C_L are the cumulative costs up to time L , $Q_L \max$ are the full qualitative benefits being derived at time L , E/C_L is the cumulative cost effectiveness calculated at time L , and L is an integer year of system life such that $0 < L \leq L \max$.

Factors of Approximation #2

(a) $\frac{R_L}{C_L}$

This is the cost factor; if R_L is greater than C_L , the system actually saves money. If the quality of performance is about the same, a system with R_L greater than C_L ($R_L/C_L > 1$) would be cost effective.

(b) $\frac{P_L}{3C_L} + \frac{\sum^L p(Q)}{L(C_L)(6)}$

This is the quality factor; if it is greater than zero, quality of performance will increase; if it is greater than one, the system is cost effective even if there are no cost savings.

Example Using Approximation #2

For R_L , P_L and C_L we have used real data from an actual analysis of a clinical laboratory system.

For $\sum^L p$ we used the following:

First year, p_1	= .132	$\sum^1 p$	= .132
Second year, p_2	= .4	$\sum^2 p$	= .532
Third year, p_3	= .875	$\sum^3 p$	= 1.407
Fourth year, p_4	= 1	$\sum^4 p$	= 2.407
Fifth year, p_5	= 1	$\sum^5 p$	= 3.407
Sixth year, p_6	= 1	$\sum^6 p$	= 4.407
Seventh year, p_7	= 1	$\sum^7 p$	= 5.407
Eighth year, p_8	= 1	$\sum^8 p$	= 6.407
Ninth year, p_9	= 1	$\sum^9 p$	= 7.407

for L we have used 1 through 9
for Q we have used the following:

1971 Management Payroll	= \$500,000
Management Effectiveness Increase	.2
Q_1	= .2(500,000) = \$100,000
1971 Non-Professional Payroll	= \$2,020,000
Operational Effectiveness Increase	.09
(accuracy, duplications, etc.)	
Q_2	= .09(2,020,000) = \$182,000
YR ₁	$Q_1 + Q_2 = Q = 280,000$

We have extrapolated Q according to the anticipated growth of payroll.

$$E/C_1 = \frac{10}{100.3} + \frac{26.8}{3(100.3)} + \frac{.132}{100.3} \frac{(280)}{6}$$

$$= .1 + .09 + .06$$

$$= .25$$

$$E/C_2 = \frac{54.8}{243.1} + \frac{139.6}{3(243.1)} + \frac{.532}{2(243.1)} \frac{(590)}{6}$$

$$= .23 + .19 + 1$$

$$= .52$$

$$E/C_3 = \frac{147}{406.6} + \frac{294.8}{3(\times 06.6)} + \frac{1.407}{3(\times 06.6)} \frac{(925)}{6}$$

$$= .32 + .24 + .17$$

$$= .73$$

Figure 6—E/C years 1-3

Cost effectiveness calculations

Figures 6 and 7 show the cost effectiveness of the system for the first three years of operation. According to our definition, if the system had a life span of 3 years, it would not be considered cost effective.

In the fourth year, there is marginal cost effectiveness. For life spans of 5 through 8 years, the cost effectiveness is good. In the ninth year it becomes excellent.

Analysis of the factors

Figure 8 summarizes the factor values which make up the total cost effectiveness.

The R factor, the cost savings, becomes marginally cost effective in 1978 and good in 1979.

This says that the system actually pays for itself in real dollar tradeoffs starting with a life span of eight years.

The P and Q factor, the qualitative benefits, do not become cost effective through the nine-year life span. This says that there must be cost savings for the system to be cost effective.

$$E/C_4 = \frac{315}{556.1} + \frac{432}{3(556.1)} + \frac{2.407}{4(556.1)} \frac{(1275)}{6}$$

$$= .56 + .26 + .23$$

$$= 1.05$$

$$E/C_5 = \frac{441}{695.3} + \frac{566}{3(695.3)} + \frac{3.407}{5(695.3)} \frac{(1630)}{6}$$

$$= .65 + .27 + .27$$

$$= 1.19$$

$$E/C_6 = \frac{622}{857.4} + \frac{712}{3(857.4)} + \frac{4.407}{6(857.4)} \frac{(1805)}{6}$$

$$= .72 + .28 + .25$$

$$= 1.25$$

$$E/C_7 = \frac{799}{901.1} + \frac{862}{3(901.1)} + \frac{5.407}{7(901.1)} \frac{(2190)}{6}$$

$$= .89 + .32 + .29$$

$$= 1.5$$

$$E/C_8 = \frac{982}{939.1} + \frac{1017}{3(939.1)} + \frac{6.407}{8(939.1)} \frac{(2580)}{6}$$

$$= 1.04 + .36 + .37$$

$$= 1.77$$

$$E/C_9 = \frac{1169}{962.1} + \frac{1177}{3(962.1)} + \frac{7.407}{9(962.1)} \frac{(2980)}{6}$$

$$= 1.21 + .41 + .42$$

$$= 2.04$$

Figure 7—E/C years 4-9

<u>YEAR</u>	<u>R FACTOR</u>	<u>P & Q FACTOR</u>	<u>E/C</u>
1	.1	.15	.25
2	.23	.29	.52
3	.32	.41	.73
4	.56	.49	1.05 marginal
5	.65	.54	1.19 good
6	.72	.53	1.25 good
7	.89	.61	1.5 good
8	1.04 marginal	.73	1.77 good
9	1.21 good	.83	2.04 excellent

Figure 8—Analysis of E/C factors

The *E/C* column adds the two factors for the resultant cost effectiveness.

Approximation #2 has been found to be most useful when the system life is not certain or when the accrual of benefits are known with more certainty as for a specific hospital area.

The application of judgmental criteria

Candidly, the relationship between the Cost Effectiveness equation and the Judgmental Criteria depends very much on the particular institution. In one case, the equation will or will not lend support to a hospital management and staff already convinced of the desirability of an automated information system and its potential to patient care and hospital efficiency.

On the other hand, such judgmental aspects could tip the scales in either direction when *E/C* approximates 1.

In any case, the cost effectiveness analysis provides the data and the insight for optimal decision making.

EVALUATION OF SYSTEM *E/C*

It is one thing to *a priori* design a cost effective system; it is another matter to determine the validity of the design by an evaluation of cost effectiveness after the system has become operational. In the main body of this paper, the author has attempted to present a case for the establishment of *E/C* parameters during the design. One of the arguments is to optimize the design. Another is to provide management with decision making tools. The third argument relates to evaluation. Without the data gathering pursuits of the design phase, the evaluation would have little with which to compare since the former "current" operation would have vanished.

The first step in the evaluation is the updating of the domain of analysis. In the great majority of the

cases this is actually an expansion. Since the design phase, new applications will have emerged through improved technology or continued enlightenment of hospital personnel as to the potential benefits of the system.

The determination of system cost effectiveness starts with establishing the criteria for evaluation. They include the benefits expected from the design as well as unpredicted benefits or negative influences.

The other major criterion is the actual system cost. The determination of system cost will be a matter of record. The hospital can establish separate cost codes for all information system equipment, personnel, facilities, material, etc.

The determination of benefits will require the same kind of analysis performed during the design study on the current system. Measurement parameters will include:

- Personnel Time—the effort required to perform the activity.
- Classification of Personnel—the task may be performed by different classifications than previously.
- Transit and Cycle Times—most related to the qualitative measurements.
- Stagnation Points—the new system can have its own bottlenecks.
- Patient Status Factors—average stay, census, etc.
- Resource Allocations—reductions or increases in personnel, equipment, material, etc.
- Qualitative Assessments—selective survey.

Essential to the evaluation is the data collection methodology and the resources utilized for data gathering.

The control data for "current" operations and their domain equivalents for "system" operations should be gathered by the regular hospital staff. The new system will impose a set of unique man-to-man and man-to-machine interactions which will require trained observers. Both the "current" and "system" operations will require test and simulation models where data describing the effect of perturbations and contingencies can be examined. Finally, acceptance and comfort of the new system will require opinion and judgmental data collections from patients, practitioners, etc.

The steps in the evaluation are exactly the same as outlined for the Absolute Cost Effectiveness.

- Cost analysis of operational system.
- Compare with current (extrapolated) system from design study—results in *R* and *P*.
- Determine qualitative benefits—*Q*.

- Determine actual costs to develop, operate and maintain the system C_s .
- Select the best approximation and calculate E/C .
- Acquire judgmental assessments.

SUMMARY

The absolute cost effectiveness analyses described in this paper result in a set of decision enhancing estimates regarding the worth of developing an information system. Implied in these data are the following criteria:

- (a) Experience in hospital operations.
- (b) Cognizance of current and anticipated information systems with hospital applicability.
- (c) An information system design philosophy and plan with associated cost estimates.
- (d) An existing library of departmental data.

Depending on the extent to which the criteria are met, an absolute cost effectiveness analysis and in-

formation plan can be performed in 2 to 4 months. In essence, the analysis provides balance sheets. It enables hospital management to attain a firmer understanding of the potential implications of the technology to the institution. It allows decision makers at every level a better picture of available design alternatives. And finally, it can provide the data to analyze the impact of the system during actual operation. In short, if used properly, cost effectiveness analysis can help take us out of the "pin the tail on the donkey" era of hospital information system design and implementation and provide the means for judging the impact of the system when in full operation.

REFERENCES

- 1 NCHS-RD-69-1
- 2 I LEARMAN
Relative cost effectiveness analysis in the evaluation and selection of hospital information systems
Presented at the Stony Brook Symposium on HSC Information Systems March 1970

Factors to be considered in computerizing a clinical chemistry department of a large city hospital

by R. MOREY, M. C. ADAMS and E. LAGA

Massachusetts Institute of Technology
Cambridge, Massachusetts

INTRODUCTION

The present biochemical testing facilities at Boston City Hospital perform about one million tests per year. To handle this workload with present personnel and equipment, the facilities must impose certain undesirable constraints on those who request tests and, in general, must limit the services which can be provided and the data which can be accumulated for subsequent analysis. These constraints ultimately imply a reduction in the health care available for patients.

The operation of the test facilities can be divided into three categories—input, processing, and output, or:

Receiving, identifying and placing samples; transcribing the tests appropriate to each sample; and assigning appropriate groups of tests to the test equipment.

Setting up equipment, assembling samples, and running tests.

Reducing data from tests; recording and storing data for future reference; and transmitting results to the requester.

The first phase of the program was to address the input and output categories relative to a specification of improvements which can be introduced into the existing test facilities of the hospital.

The essential objective of the program was to provide, in a relatively short period of effort, specification for a clinical chemistry operation of significantly increased effectiveness. The specification was to be applicable to existing clinical chemistry operation and consistent with a long-term program of continued improvement in service, reliability and cost savings. Thus, the short-term consideration must be consistent with the long-term goals.

To accomplish the above objective, a comprehensive set of requirements was established at the outset of the proposed program in conjunction with the Boston City Hospital. These requirements were based on an intensive analysis of the present clinical chemistry operations and took into account current and projected user needs, accumulated experience in the hospital, and experience obtained in operating comparable systems.

Concurrent with the development of requirements, the operational flow in the present facility was delineated and cost data collected and analyzed in relation to the requirements established to determine future needs. Appropriate existing systems and equipment were evaluated relative to needs and recommendations based upon available funding prepared for use as a basis for implementation.

In relation to the approach outlined, managing the "throughput" was a major aspect of the program. The operation had to be designed to handle a normal stream of tests as well as to respond to "demand" tests. The testing process had to be rigidly controlled to assure reliable results. The entire operation was under constant supervision and monitoring; including feedback controls, abnormality signalling, test-interference protection, and comparison of results against known standards.

The hospital biochemical laboratory nowadays performs the majority of the requested analyses by automated equipment. Except for specialized tests, and in small hospitals, manual methods have been superseded by automated and semi-automated methods. Blood chemistry tests are being called out in an ever-increasing manner to aid diagnosis and to monitor a patient's condition during his stay in a hospital. When an analysis becomes very complex, automation is often the only means to develop a consistent and repeatable analyses.

The hospital laboratory staff must translate chart values typically obtained on a recorder into values expressed in concentration per unit volume. This requires mathematical or graphical manipulation and interpretation including calculation of base line drift in which errors can occur due to fatigue or inattention. The staff must prepare daily log sheets, laboratory summary and statistical reports, and quality control reports and the possibility of errors always exists.

These procedures are all clearly within the capability of currently available computing systems. A solution would be to utilize an on-line system (hardware/software) that provides the above services during the time of actual specimen processing.

The laboratory at BCH is inundated with large numbers of specimens every morning requiring multiple sorting to prepare them for testing by automatic analytical instruments. There is always the possibility present of lost identification of the specimen, the test and even of the patient.

The tests themselves are now automated and this is the only way a laboratory can keep up with the volume of tests but another imbalance is caused by non-automation of the data input and output and record keeping causing a reduction in the advantage obtained from the technological advances in instrumentation.

Lamé in an article in *Laboratory Medicine*, November, 1970, states that a survey of ten clinical laboratories concluded that a minimum of 20 percent and more likely 30 percent of the technologist's time was being spent in performing clerical duties. Our study at BCH indicated that the 30 percent figure is a good estimate. Writing test reports, keeping log books, labeling vacutainers and test tubes, preparing work sheets, hand and slide rule calculations, billing, updating patient files, answering telephone enquiries, are but a few of the clerical duties being performed in a typical clinical laboratory.

It is evident that the solution to this problem may be found by examining other industrial and research operations where the introduction of data processing and the use of a dedicated on-line computer has been most beneficial.

The doctor's prime requirement is a supply of up-to-date information concerning his patients. Test requests are initiated by the doctors who desire clear, concise laboratory reports of the results obtained. To provide this capability using a laboratory data handling system, the laboratory staff must be able to enter information on test results into the system. Also queries as to the status of on-line instruments and the contents of patient files must be entered. Similarly, test procedures must be capable of modification as procedures are changed in the normal course of laboratory develop-

ment. In addition, routine reporting of test results on each patient and documentation of laboratory work performed can be initiated by the system itself or may be requested by the laboratory operating staff.

A computer will function well only in an organized system. A comprehensive system analysis such as was carried out for BCH and is delineated later in this report is an essential prerequisite prior to computerization to determine the most efficient procedures to use.

The disadvantages of computerization are concerned with cost, the transition period and the development of operating procedures in case the computer system goes down due either to a power failure or some mechanical or software problem.

Another disadvantage in computerizing the clinical laboratory is the tendency to treat computerization as an end, rather than a means.

It is possible in the interim stages that a computerized system will add to the cost rather than reducing the overall cost of the laboratory but in general, hospitals which have automated, have found this to be short range objection. There may also be resentment in that the laboratory will be dependant on outside vendor personnel from the supplier of the system. One solution would be to use the hospital electronic data processing personnel as an interface between the laboratory and the system vendor or even to utilize a programmer on the laboratory staff. When laboratory personnel understand the operation and run their own computer system, feelings of distrust and antagonism should be overcome.

One essential requirement for any laboratory system is that it must have a high degree of reliability to consider the results significant and be worth the expense financially and personnel-wise.

A projected use for the computer once the reliability of the hardware and software are verified is for the use of the system in differential diagnosis. The computer will be able to extend and provide backup for the physician's medical decision-making ability. It could be used to generate such differential diagnosis on the basis of the laboratory data and to be able to generate lists of additional tests to be done to generate more specific diagnoses. Such adaptation of computer science would affect both the quality and cost of health care.

SYSTEMS ANALYSIS OF EXISTING OPERATION AT BCH (1970)

A flow chart indicating the major operational steps in the Clinical Biochemistry Department of the Boston City Hospital (BCH) is shown in Figure 1. This in-

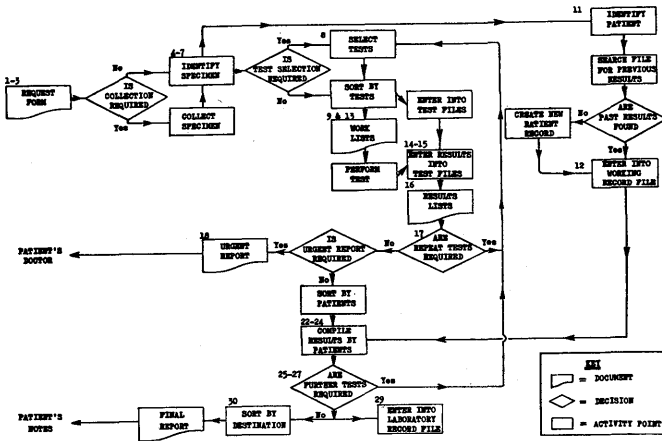


Figure 1—Flow chart of operations

cludes (a) test requesting, and specimen collection which are performed by ward personnel; (b) accessioning of samples and of request forms, preparation of worklists and result lists, and administrative logging, which are performed by technical and clerical staff of the laboratory; (c) analysis, computation of results and transcription of results to report forms which are performed by the analysts; (d) sorting of results by patient and wards and creation of alphabetical long-term files which is done by the clerical staff of the laboratory.

In the following Tables and Graphs the present and future workload as it relates to in- and out-patient services is considered within-laboratory work scheduling of the staff and its analytical performance and some aspects of budgeting. Next will be discussed some of the discrete operations involved in the requesting, accessioning, analyzing and data processing steps of the

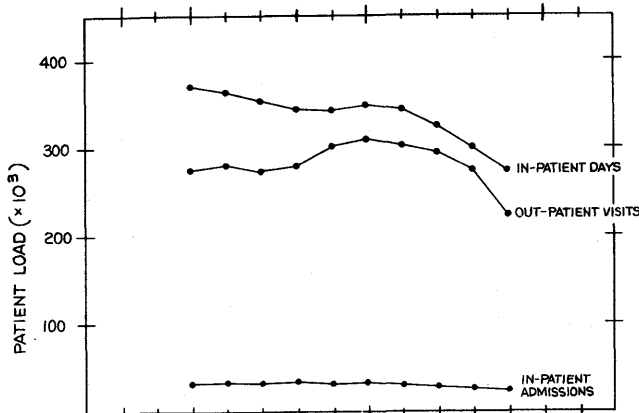


Figure 2—BCH-CC: Annual patient load (1960-1970)

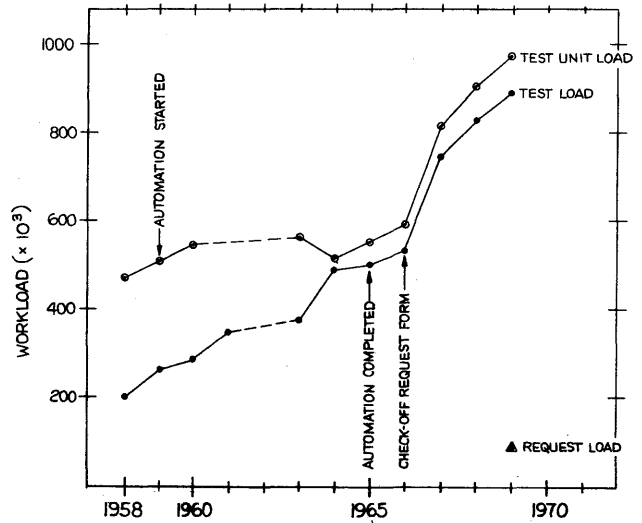


Figure 3—BCH-CC: Annual chemistry load (1958-1970)

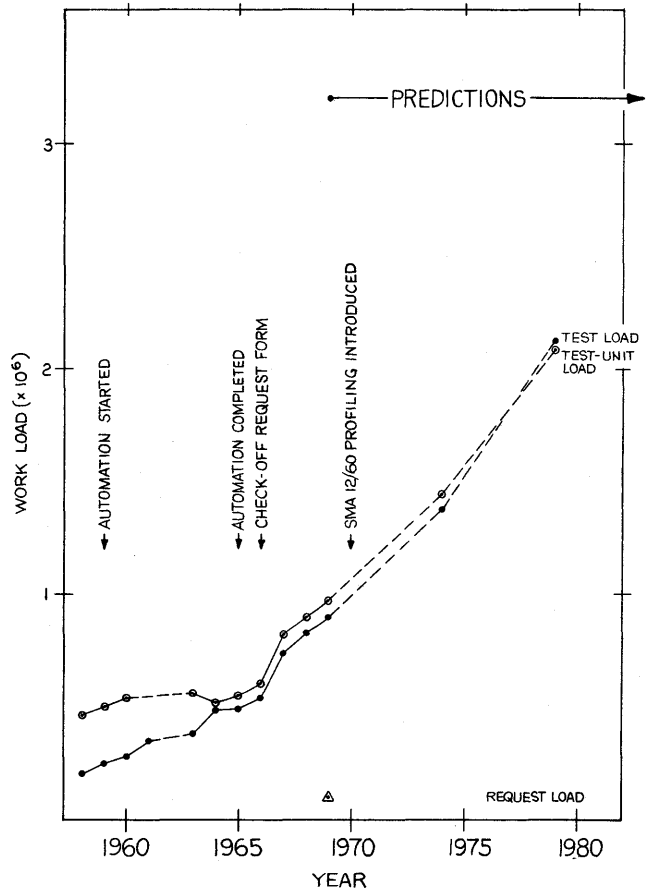


Figure 4—BCH-CC: Annual chemistry load forecast (1969-70)

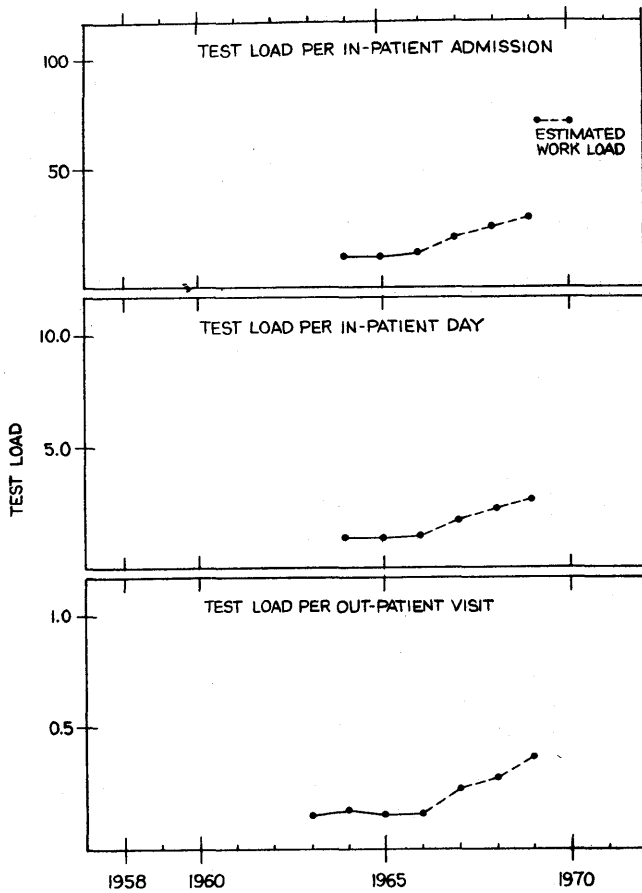


Figure 5—BCH-CC: Annual test load, by patient (1958-70)

entire operation. This will be followed by a discussion of characteristics of hardware, software and peripheral equipment currently available from different computer manufacturers. Alternative proposals for introduction of a computer-assisted system for on line monitoring of Auto-Analyzers and for preparation of cumulative reports will conclude this section.

The recent decrease in utilization of both in- and out-patient facilities of BCH is shown in Figure 2. This has been accompanied by a steadily increasing request load, test load and/or test-unit load,* (Figure 3) in Clinical Biochemistry. If the test load keeps rising at the current rate, there will be an increase from the present 1 million tests per year to about 1,500,000 tests per year by 1975 and to 2,000,000 tests per year by 1978-79 (Figure 4). Test load for chemistry per out-patient visit is still about $\frac{1}{10}$ less than in-patient day. The latter is expected to rise faster in the next decade than the former (Figure 5). A breakdown of the

* Test units are tests weighted by U.S. Veterans Administration AMIS Test Weighting System.

chemistry load by test for the 1958-70 period and its breakdown in automated and non-automated tests shows an increasing demand for tests and the increasing use of automated tests during this period (Figure 6). About 3 automated tests are responsible for 25 percent of the load, about 6 for 50 percent, about 9 for 75 percent and about 13 for 90 percent (Figure 7). This is indicative of a high volume operation with a rather limited diversification in terms of tests offered. The latter situation obviously creates a more acute demand for computer assistance than a more diversified, low-volume operation.

Monthly variations in test load for day-, evening-, and night-shifts in 1970 are shown in Figure 8. By the end of 1970, about 10 percent of the work load was performed on the SMA 12/60s. Workload on Mondays occasionally is twice as large as on any other single day of the week. Average test load per request is from 8 to 10 depending on day of week and month of the year

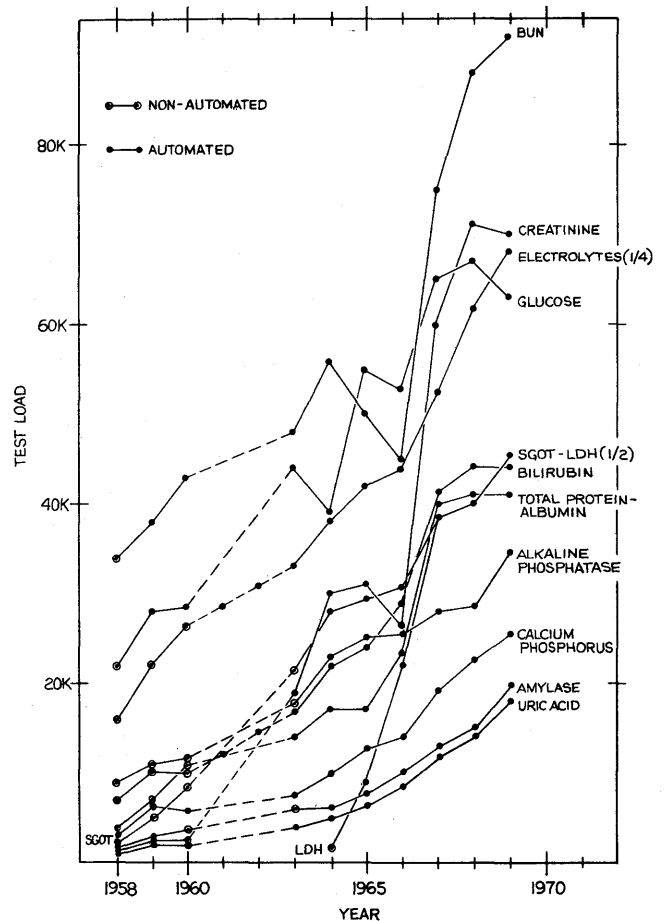


Figure 6—BCH-CC: Annual automated test load, by test (1958-70)

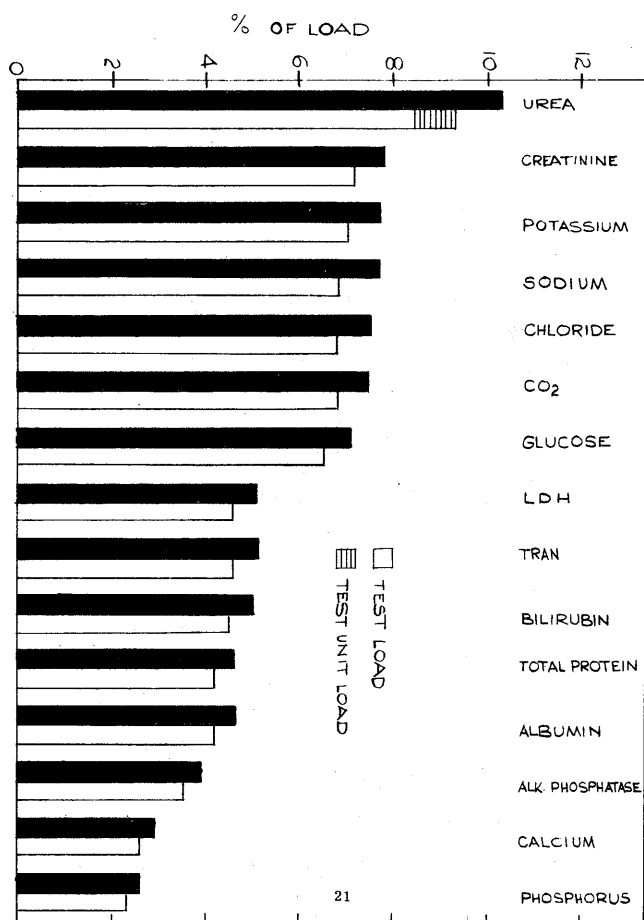


Figure 7A—BCH-CC: Fractional test load ranking, by test (1969)

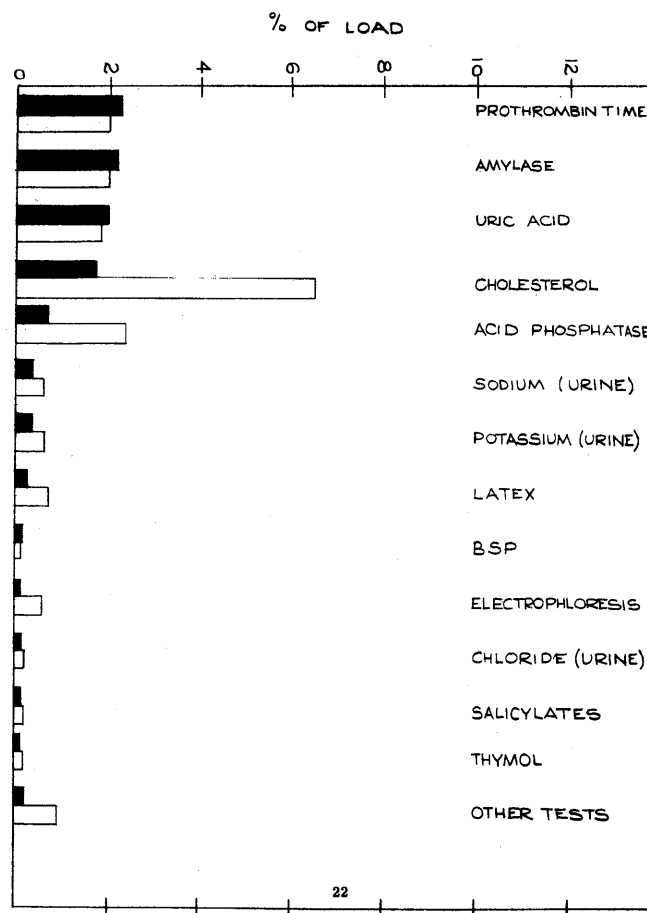


Figure 7B—BCH-CC: Fractional test load ranking, by test (1969)

(Table I). The Medical Services are responsible for about 50 percent of requests and about 60 percent of tests performed. The remainder of the requests originate from the Surgical Services, Clinics, and Outpatient Services (Figure 9).

Within-laboratory work scheduling for analysis, administrative personnel and supervision is shown on a time basis for different shifts and work assignments in Figure 10. In total, about 96 analytical man-hours are available per day. Two-thirds of these are spent on true analytical work and one-third on clerical jobs. About 32 man-hours per day are available for administrative and supervisory assignments.

At a current 95 percent level of test automation in the laboratory, a total of 14 analysts performed about 70,000 tests per analyst-year, which is above the national average for hospital laboratory performance.

From the available budget figures, it appears that the operational cost for the annual test load of about

TABLE I
BCH - DAILY CHEMISTRY TEST LOAD* (1970)

	All Day	WEEK-DAY			WEEK-END	
		Day Routines	Day Emergencies	Night Emergencies	Saturday	Sunday
SAMPLE LOAD						
Average	406.7	298	54.5	54.2	136.2	96.4
Minimum	291	210	40	41	44	68
Maximum	656	512	72	72	183	170
Average, as % all-day	100	73.27	13.40	13.32	-	-
TEST LOAD						
Average	4,052	3,498	292	262	712	405
Minimum	2,396	1,998	208	190	511	324
Maximum	5,903	5,186	369	348	879	566
Average, as % all-day	100	86.33	7.20	6.47	-	-
AVERAGE TEST LOAD PER SAMPLE						
	9.96	11.73	5.35	4.83	5.23	4.20

*March 1970

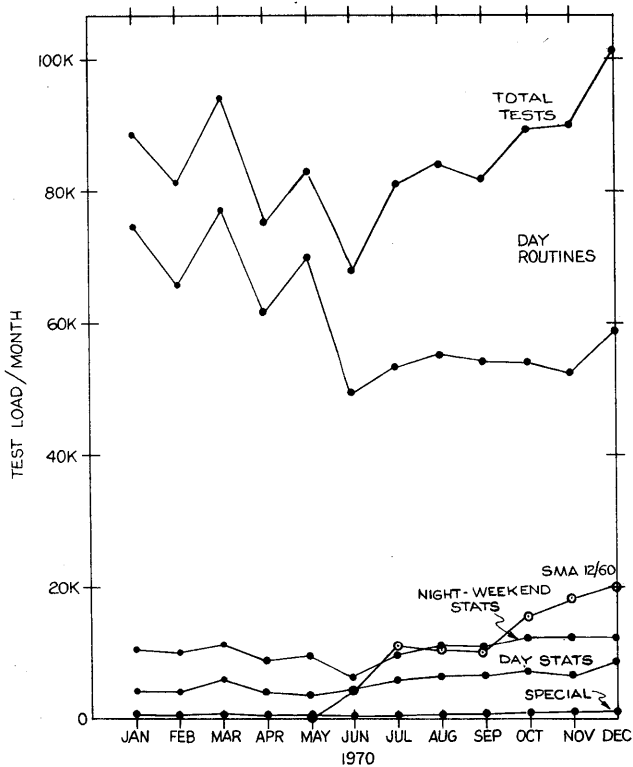


Figure 8—BCH-CC: Monthly chemistry test load (1970)

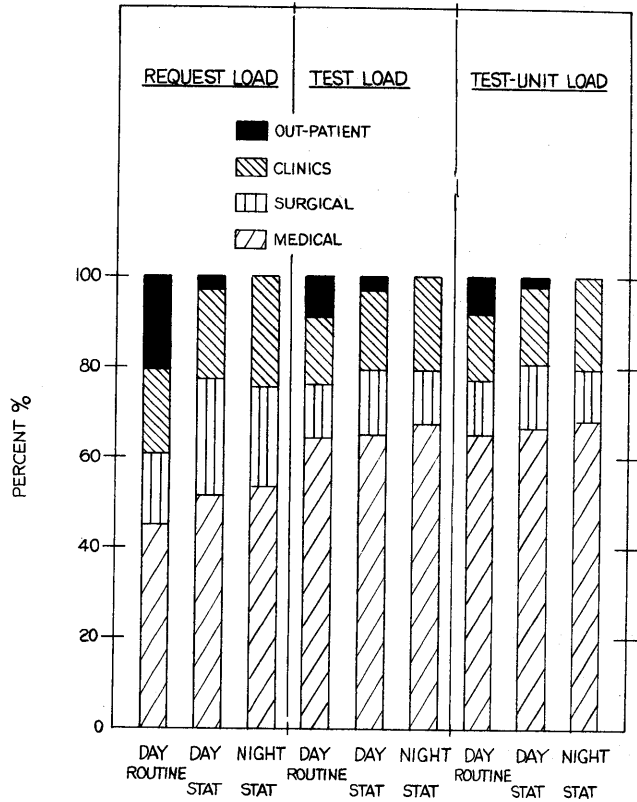


Figure 9—BCH-CC: Fractional daily chemistry load, by service (1970)

900,000 tests in 1969 was about \$250,000. Average cost per test amounted therefore to about \$.28.

Test requesting is done by using three types of request forms (Table II). This is a check-off form with an Addressograph-Multigraph card embossing system for patient identification. This type of form and other factors determine the pattern of test requesting. Only about 20 percent of test requests are for single tests or for test-group run in parallel on multichannel analyzers. The majority of test requests (75 percent) require sample splitting for analysis in a number of single or multi-channel analyzers. The frequency of requests for single tests and test groups indicates the small number of tests which do not require sample splitting. The frequency of requests for single tests or test groups for the same tests requested in combination with other test groups indicates the variety of permitted test group combinations. Figure 11 shows the frequency of tests requested per request, the mean being about eight tests per request. This list may serve as a guideline for selection of those requests which would be more efficiently handled by a 12- or 16-channel analyzer, thereby eliminating time and errors involved in multiple sample-splitting.

Accessioning of request cards and samples for routine

test starts with the assignment of an accession number to both card and tubes. Tubes are placed in racks on a long bench on top of a sheet of paper. Accession number and requested tests for the accession is written down on the sheet of paper. After centrifugation, serum is

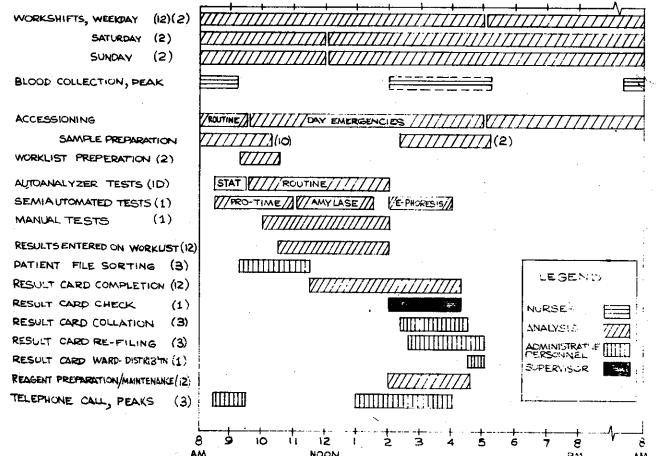


Figure 10—BCH-CC: Work scheduling by staff (1970)

1 BIOCHEM. DATE: _____

PREVIOUS CHEMISTRY: YES NO DATE: _____

NAME: _____ HOSP #: _____ WARD: _____ SERVICE: _____

DIAGNOSIS: _____

USE ADDRESSOGRAPH PLATE OR WRITE LEGIBLY!!!

CODE	TEST	RESULTS	CODE	TEST	RESULTS	CODE	TEST	RESULTS
2086	BUN		2057	BR TOT		2242	LATEX	
2342	NA		2058	T & D		2079	BSP	
2298	K		2303	TP		2330	SALY	
2107	CL		2082	ALB		2091	CA	
2083	CO ₂		2287	ALP		2289	P	
2024	AMY		2286	ACP		2377	TRANS	
2351	BS		2110	CHOL		2124	CREAT	
2391	UA		2373	TT		2087	LDH	

2 BIOCHEM. DATE: _____

PREVIOUS CHEMISTRY: YES NO DATE: _____

NAME: _____ HOSP #: _____ WARD: _____ SERVICE: _____

DIAGNOSIS: _____

USE ADDRESSOGRAPH PLATE OR WRITE LEGIBLY!!!

CODE	TEST	RESULTS	CODE	TEST	RESULTS
2306	PT		2085	SWEAT NA	
2160	ELECTROPHORESIS		2084	pH	
2109	CHOLESTROL AND ESTERS		2163	FIBRINOGEN	
2105	CF		2118	CONGO RED	
2354	SULPHA		OTHER (SPECIFY):		

3 EMER. BIOCHEM. DATE: _____

PREVIOUS CHEMISTRY: YES NO DATE: _____

NAME: _____ HOSP #: _____ WARD: _____ SERVICE: _____

DIAGNOSIS: _____

CODE	TEST	RESULTS	CODE	TEST	RESULTS	CODE	TEST	RESULTS
2086	BUN		2083	CO ₂		2057	BR TOT	
2342	NA		2351	BS		2058	T & D	
2298	K		2024	AMY		2306	PT	
2107	CL		2079	BSP		2330	SALY	

NOTE: EMERGENCY REPORTS WILL BE TELEPHONED TO SERVICES UPON COMPLETION. NAME AND SERVICE SHOULD APPEAR ON SPECIMEN AS WELL AS ON THIS FORM.

REQUESTED BY: _____ TELEPHONE: _____

DP FORM 1703

TABLE II
BCH: CHEMISTRY INSTRUMENTATION (1970)

	Instrument	Sampler	Recorder	Dialysis	Samples Per Hr.	% Repeats	Criteria for Repeats
1. Bun	AA-color	1	1		90	15-20	>150
2. Creatinine	AA-color	1	1		90	15-20	> 15
3. Glucose	AA-color	1	1		90	5-10	<50, >300
4. Uric Acid	AA-color	1	1		70	5-10	> 12
5. Sodium	AA-flame		2		70	5-10	NA, C1
6. Potassium	AA-flame		2		70	5-10	
7. Chloride	AA-color	1	1		70	5-10	Discrepancy
8. Carbon Dioxide	AA-color		2		70	5-10	
9. Bilirubin	AA-color		1		120	10-15	> 7
10. Total Protein	AA-color	1	1		120	5-10	<5 > 8
11. Albumin	AA-color		2		120	5-10	<2 > 5
12. Alkaline Phosph.	AA-color	1	1		70	5-10	> 8
13. LDM	AA-color		1		70	15-20	>1125
14. SGOT	AA-color	1	1		70	15-20	> 160
15. Amylase	AA-color	1	1		70	10-15	>1,400
16. Calcium	AA-color		2		70	15-20	<85 >12.5
17. Phosphorus	AA-color	1	2		70	15-20	<20 >10
18. Profile, Chemical	SMA 12/50	1	1		60	5	-
19. Prothrombin Time	Fibrometer			NA	60	5	-
20. Sodium Urine	IL, flame		2		180	5	-
21. Potassium, Urine	IL, flame	1	2		180	5	-
22. Trace Metals	PE, AAS 303	1	1		-	-	-
23. Cholesterol	Coleman, Jr. III				-	5	-
24. Acid Phosphatase	Coleman, Color				-	5	-
25. Electrophoresis	BM - Analytrol - GM-Digiscreeen				-	240	-

manually transferred to a serum tube on which the accession number has been transcribed. Sample splitting from serum tube to Auto-Analyzer cups is done manually, using work lists to select serum samples and to list the sequential position number of the sample on the Auto-Analyzer sample tray for each accession number. Results are collated after the Auto-Analyzer run from recorder chart and final results are manually computed from the raw data. The obtained results are matched to cup number, next to accession number and finally to the patient's name before final transcription on the report form.

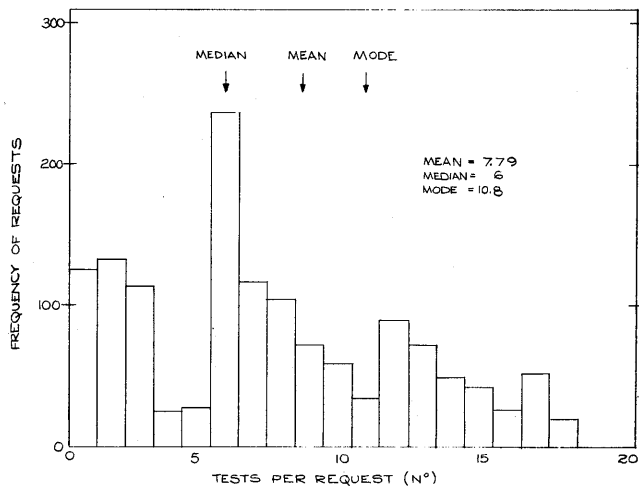


Figure 11—BCH-CC: Frequency of multiple test requests (1970)

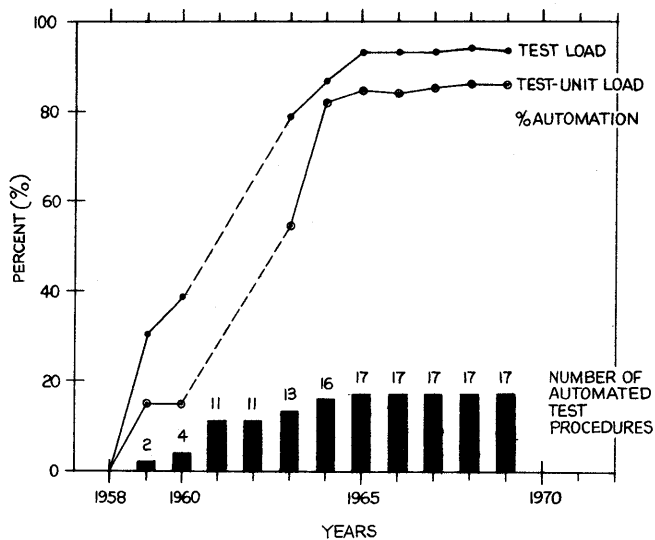


Figure 12—BCH-CC: Percentage automation of test load (1958-70)

Table III lists the currently available instrumentation in the laboratory. Eighteen analog Auto-Analyzer channels are potential candidates to be multiplexed in an on-line computer-assisted data handling system. The sampling rates for different tests are indicated. Also shown is the percentage of repeat tests and the criteria used for repeating an analysis. Figure 12 indicates the number of automated tests procedures and the percentage of test and test unit-load which is automated.

For routine specimens, the time delay between receipt of the sample and result leaving the laboratory is about eight hours. One report form goes to the patient's chart and one to the Billing Office. Long term files for both

in- and out-patients are kept in the laboratory using another duplicate of the report forms filed alphabetically. File retrieval is done manually. Requests for retrospective cumulative retrieval have obviously to be limited because of limited available personnel to perform the task. No separation exists in the file structure between active and inactive patient-files.

DESCRIPTION OF AVAILABLE LABORATORY COMPUTER SYSTEMS (1970)

A standard procedure was established for evaluating the various turnkey systems available at present. All

TABLE IV

HARDWARE CHARACTERISTICS OF 12 COMPUTER SYSTEMS (1970)

	BSI Clin- Data	Con- comp I	DEC Clin- Lab-12	DNA CLS	Info- tronics CL II	IBM 360/40 Batch	IBM 1130	IBM 1800	IBM 1080	IBM Syst/17	Medi- tech	Spear Class 300-B
Turn-Key System	yes	yes	yes	yes	yes	no	no	no	no	yes	yes	yes
Computer in Lab	yes	yes	yes	yes	yes	no	yes	yes	no	yes	no	yes
Purchase Price (\$)	188,600	165,000	151,083	175,000	175,000	500,000	90,000	250,000	24,000			165,000
Rental Price\$(Mo.)	4,350						2,000	4,400-	600.	10,000		
Maint. Contract (\$) after first year			550									
No. Autoanalyzer Channels	16	32	24	16	40	NA*	NA	NA	NA	NA		28
Time to Complete Installation (Mo.)	9	3	4-6	4	6	12	4	4	4	12	2	4
Computer	PDP-8	Micro- Syst. 800	PDP-12	2700 Ray- theon	PDP-12	360/ 40	1130	1800	1080	Syst/ 7	PDP-12	Spear
Operating Installations (Jan. 1971)	13	0	2	6	1	4	5	7	7	0	1	13
Total Responsibility by Vendor	yes	yes	yes	yes	yes	no	no	no	no		yes	yes
Factory On-Job Training	yes	yes	yes	yes	yes	no	no	no	no	yes	no	yes
Teletype Terminals	yes	only	yes	yes	only							
Specific Console Terminals	yes	no	yes	yes	no	no	no	no	no	NA		
CRT Terminal	no	no	yes	no	no	no	no	no	no	NA	yes	yes
Power Requirement												
BTU Generated by equipment/hr.												
User Group Available for Consultation	yes	no	no	no	no	no	no	no	no	no	no	yes

* Standard Configuration

TABLE V

SOFTWARE CHARACTERISTICS OF 12 COMPUTER SYSTEMS* (1970)

	BSL Clin- Data	Con- Comp I	Dec Clin- Lab 12	DNA CLS	Info- tronics CL II	IBM 360/40 Batch	IBM 1130	IBM 1800	IBM 1080	IBM System 7	Medi- Tech	Spear Class 300-B
Patient Directory	yes	no	yes	yes	no	no	no	yes	yes		no	no
Specimen Collection	yes	yes	no	yes	no	no	no	yes	no		no	yes
Test Work List	yes	yes	yes	yes	yes	no	no	yes	yes		yes	yes
Load List	yes	yes	no	no	yes	no	no	yes	no		no	no
Ward Report	yes	yes	yes	yes	yes	no	no	yes	yes		no	yes
Day Report	yes	yes	yes	yes	yes	no	no	yes	yes		no	yes
Query Report	yes	no	yes	yes	yes	no	no	yes	no		yes	yes
Abnormal Value Report	no	no	no	no	no	no	no	no	no		yes	yes
Unfinished Procedures Rpt	yes	no	yes	yes	no	no	no	no	no		no	yes
Quality Control Report	yes	no	yes	yes	no	no	no	no	no		no	no
Daily Statistical Report	yes	no	yes	no	yes	no	no	no	no		no	yes
Billing Report	yes	yes	yes	no	no	no	no	yes	no		yes	yes
Cumulative Rpt. for Days	yes 7	no -	yes 7	yes 7	no -	no -	no -	yes 7	no -		yes 6	yes 16
Management Report	yes	no	no	yes	no	no	no	no	no		no	no
EDP Compatibility	no	no	no	no	yes	no	no	yes	no		no	?
IBM Retrieval Programs	no	no	no	no	no	no	no	no	no		no	
Sample Identification Sys.	no	no	no	no	no	no	no	no	no		no	no
Computer Manufacturer acts as Consultant	no	no	no	no	no	no	no	no	no	no	no	yes

*Standard Configuration

known vendors in the field were evaluated together with other computer manufacturers who have shown an interest in entering the clinical chemical computerization field.

The initial step was to obtain relevant literature on the systems. A preliminary evaluation was made after reading this information to determine the applicability of such systems to the requirements of BCH. An important segment of this evaluation was to ensure that the companies had the proper software to meet all or most of the BCH requirements. This included Day Reports, Cumulative Reports, Patient File, Quality Control, Work Lists, and other types of miscellaneous

programs available to support a clinical laboratory operation.

The second step after familiarization with the literature was to request a representative from the company to come and give a presentation of their system. Generally this presentation was given by a technical person from the Bio-Medical Department rather than a salesman. The presentations augmented the technical data previously sent, showing a typical hospital arrangement, the consoles available and typical format of chemical and patient data output. There was a discrepancy in that in certain cases the literature stated that certain software was available but the

TABLE VI
INPUT/OUTPUT/STORAGE CHARACTERISTICS OF 12 COMPUTER SYSTEMS (1970)

	BSL Clin-Data	Concomp I	DEC Clin-Lab	DNA 12 CLS	Info- tronics CL II	IBM 360/40 Batch	IBM 1130	IBM 1800	IBM 1080	IBM Syst/17	Medi- tech	Spec Class 300-B
I. Data Input Equipment												
1. Mark Sense Cards	yes	no	yes	no	no	no	no	yes	no	no	no	yes
2. Port-a-Punch Cards	no	yes	no	no	no	yes	yes	yes	yes	no	no	yes
3. Bar-Coded Cards	no	no	no	no	no	no	no	no	no	no	no	no
4. Keyboard Terminal	yes	no	no	no	no	no	yes	no	no	no	no	no
5. Key-Mat. Terminal	no	no	no	no	no	yes	yes	yes	yes	yes	no	no
6. CRT at CPU	yes	no	yes	no	no	no	no	no	no	no	no	yes
7. CRT-Remote	no	no	yes	no	no	no	no	no	no	no	yes	yes
8. Teletype, at CRU	yes	yes	yes	yes	yes	no	no	no	no	no	no	yes
9. Teletype, Terminal	no	yes	yes	no	yes	no	no	no	no	no	no	no
10. Specific Purp. Term.	yes	no	no	yes	no	no	no	no	no	no	no	no
II. Data Output Equipment												
1. Teletype	yes	yes	yes	yes	yes	no	no	no	no	no	no	no
2. Kleinschmidt Printer	yes	no	no	no	no	no	no	no	no	no	no	yes
3. Line Printer	yes	yes	yes	yes	no	yes	yes	yes	yes	yes	yes	yes
4. Plotter	no	no	no	no	no	no	no	no	no	no	no	no
III. Data Storage												
1. DEC-Tape	yes	no	yes	no	yes	no	no	no	no	no	yes	yes
2. IBM Tape	no	yes	no	no	no	yes	yes	yes	yes	yes	no	yes
3. Disc	yes	yes	yes	no	yes	yes	yes	yes	yes	yes	yes	yes

presentation indicated that some programs were still under active development. This is understandable because the verification of software without an expensive in-plant simulation for the verification of programs is extremely difficult. As proof, the Apollo flight programs are still finding bugs after six years and millions of dollars worth of verification testing.

The vendors were carefully questioned as to the operating characteristics of the computer including air conditioning requirements, the preventive procedures maintenance and the response time for emergency service.

The third step in our evaluation was to try to visit representative systems in a hospital. This was difficult as in most cases the systems were in process of being installed, they were of an early configuration that was not representative of later designs or they were installed but not operational.

CONCLUSIONS

In general most of the systems were similar as far as cost and type of software available. The major factor that

separated the companies was the number of working systems which they had in the field which is directly proportional to the length of time they have been marketing systems. The number of systems operating is thus not an efficient means of evaluating the companies. Evaluation therefore must be based on the service that the individual companies will provide with their equipment.

A general survey of 12 computer systems available in 1970 with respect to hardware characteristics (Table IV), the accompanying software (Table V) and the associated input-output storage devices (Table VI) has been compiled.

AVAILABLE LABORATORY COMPUTER SYSTEMS (1970)

A form was circulated to hospitals with known installed systems. A request was made for their comments on the equipment as installed in the Clinical Laboratory and it was emphasized that this was in no way to be connected with the user by name.

An analysis was made on the returns for the various systems.

It was noted that the longer a particular system has been marketed the higher the probability that deficiencies have been rectified. It is only when major changes are incorporated such as substituting disk storage for tape storage that major difficulties are found by a particular user.

Also the efficiency with which a particular system is used is bound up with the attitude of the technicians and the dedication with which the laboratory director endeavors to integrate the system into his operation.

It was shown that it is essential that a systems analysis as has been performed at BCH be carried out. The type of hospital, i.e., teaching versus non-teaching, also is important in estimating the number of tests per patient that are to be carried. A further observation is that the time for complete integration of the computerized system into the work schedule of the laboratory can be excessively long. Again this is probably a reflection of software efficiency and technician acceptability. The down time is noted to be excessive in a number of cases which indicates that the laboratory must always maintain the capability of switching to a manual mode when this need arises.

ACCESSIONING AND POSITIVE SAMPLE IDENTIFICATION SYSTEMS (1970)

At this time there is no positive sample identification system which would be practical for Boston City Hospital. This is due to the fact that Boston City does not have a collection team but leaves the collecting of the blood up to the individual ward nurses. This would require a large investment at each collecting station for equipment to mark the samples. An embossed card can be used for sample I.D., reference Figure 13. This can be done in the following way: the form is imprinted with patient's name which is human readable, and patient's I.D. and ward number in machine readable form. The blood is drawn and a request for test is attached to the vacutainer. These are now sent to the lab where the technician will assign an acquisition number and mark in the number on the mark sense card. The container is placed under the assigned number on the work table and the cards are collected and fed through the card reader. The computer then makes up the work list which tells the technician which specimen, by acquisition number, is to be placed in which location in the sample loader. Upon completion of the tests the computer will type out the ward report which will have the patient's I.D. number and the test results. The reports are then sent to the wards where the nurse will assign the test to

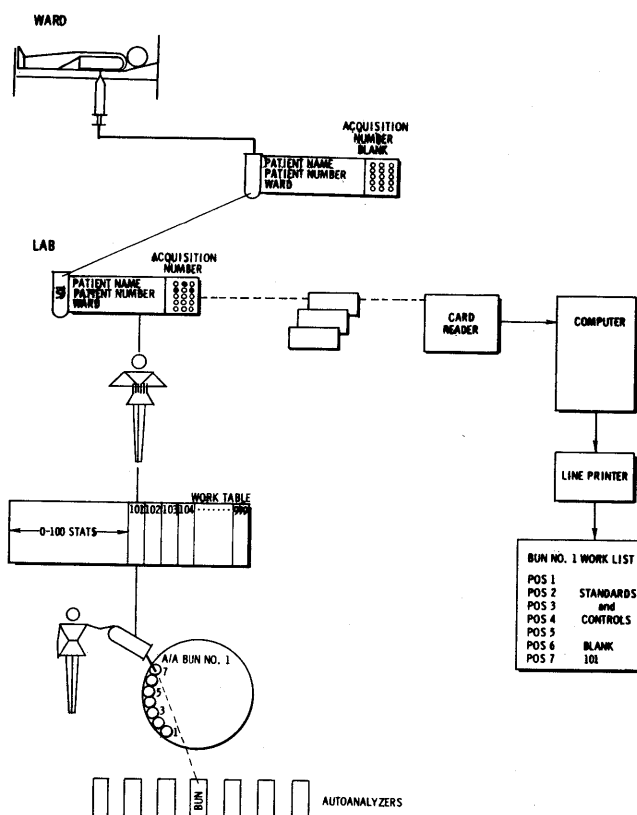


Figure 13—Sample accessioning and identification scheme

the patients according to their patient I.D. With this method all queries will be by patient I.D. number only. There will be no flagging of out of normal limits because patient's age and sex are unknown. If working without the patient's name is satisfactory, this method is a very fast and efficient way of getting positive simple identification. If the patient's name is required the technician can type in the patient's data. On most systems about four entries a minute can be made. Another way of accomplishing this is to have the wards send a copy of their collection lists the night before so that the people on the night shift in the lab could update the patient's data for the following day. This would take care of all patients except the out-patient and stats which would have to be entered upon receiving the blood sample in the lab. Tables VII and VIII represent a survey taken of available schemes by different manufacturers.

RECOMMENDATIONS

There are several ways to computerize the lab at BCH. These depend on the degree of involvement of the

TABLE VII SAMPLE ACCESSIONING SYSTEM (1970)

	American Science and Engineering	Spear	Technicon AA II Idee	IBM [®]	Vickers M300	Identicon [®] Identiscan TM100	American Cyanamid DMS System	Dupont Automatic Clinical Analyzer	Damon [®] Engineering Microtainer
Positive Patient Identification Tab	Armband with Patient Hospital I. D. No./Bar Code: Bar Code transcribed through transcription device on vacutainer heat sensitive label	None	None	None	None	None	None	None	NIA
Bedside Container	Vacutainer	Vacutainer	Vacutainer	Vacutainer	Special Flat 1.5 ml vial	Vacutainer	Vacutainer	Vacutainer	Microtainer
Request Card	In-House Card	Under development	Special combined request/preprinted Idee label (Bar code, peel-off)	Special combined request card/tear-off prepunched tab	In-House Card	In-House Card	In-House Card	In-House Card	NIA
Sample Container Code	Electrostatically printed on Vacutainer label from Armband	Under development	Vertical prepunched bar code on Idee label	Vertical Hollerith, prepunched on tab card	Vertical Hollerith, post-punched on label fixed to special vial	Retro-reflective horizontal bar code on label	Fluorescent vertical bar code on label	NIA	NIA
Sample Accessioning	Hospital No Bar Code Transcribed from Vacutainer Label to transfer tube AA cup via transcription device	Under development	Preprinted Idee label entered in Log Book one on transfer tube/AA cup	Prepunched tab card attached to transfer tube/AA cup	Sequential accession number, post-punched on label fixed to vial	Under development	Under development	Sequential accession number entered on identification card attached to special serum cup	Sequential accession number entered on identification card attached to microtainer
Request Card Accessioning	NA	HP Card Reader	IBM Card Reader	Pre-accessioning via telephone/computer	NA	Under development	Under development	NA	NIA
Centrifugation	Vacutainer-Holder	Vacutainer-Holder	Vacutainer-Holder	Vacutainer-Holder	Special vial holder	Under development	Under development	Vacutainer-Holder	NIA
Serum Transfer System	Manual	Manual	Manual	Computer Controller sequential serum aspiration from original sample tube in single/dual channel Auto Analyzers from continuous belt. Optical control of sampling depth	Sequential serum aspiration from original sample tube. Optical control of sampling depth	Under development	Under development	Manual	NIA
Test Container Code	AA cup carrying electrostatically imprinted Bar Code	None	AA cup carrying Idee label	Original sample container carrying prepunched tab number	Original sample container carrying postpunched number	Under development	Under development	Dupont sample cup with attached identification card	NIA
Sampler	Technicon AA sampler II, III (40 positions)	Technicon AA sampler II, III (40 positions)	Technicon AA sampler III (40 positions)	No sampler needed	Vickers M300 sampler (300 positions)	Under development	Under development	Linear sample/reagent rack set input tray	NIA
Scanner/Decoder	Reflection Scanner	Under development	Reflection Scanner	Transmission Scanner	Transmission Scanner	Reflection Scanner	UV-fluorescence Scanner	Photographic reproduction of ID information from ID card attached to sample to report form	NIA
Digits	6	NIA	6	6	6	10	10	NA	NIA

[®] Under Development
[®] Monmouth Modification
 NIA = No Information Available

laboratory with the installation and the projected utilization of the equipment after installation. Do they want a turnkey system, like another instrument, which will help them in their existing workload or do they hope to expand and use the computer system for other functions? Reference Figure 14.

If a turnkey system only is required our recommendation is that they proceed with plans to purchase a system from either Digital Equipment or SPEAR.

If the lab plans on imaginative use of the computer system, it is felt that a Bio-Engineer should be hired. With this person on the staff, the use of the equipment would be greatly enhanced. This technical person would assist the director in whatever he might require in the areas of new test procedures or new programs. An interesting development along this line is noted by Doctor Lamé in Laboratory Medicine, November, 1970,

TABLE VIII TEST REQUEST AND SAMPLE IDENTIFICATION SYSTEM

	Addressograph Multigraph	H-P Hewlett-Packard	IBM	COPAC	Card Image	Telephone Activated - Request Sys.
Code	Bar Code	Dial Code	Hollerith Pch Mark Port-a-Punch	Hollerith	NA	NA
Installations	BCH	BSL	Spear	J. C. Med. Ctr. S. F., Calif.	Sanders/Kaiser, Lockheed/ Mayo Clinic	Monmouth County Hospi
Admissions Plate Punch	6400	NA	NA	Templa Punch 501	Sanders Plate Punch	NA
Nursing Station Punch	A&M 12-45			Templa Punch 501T	NA	NA
Service Area Punch				Selecta Punch 5082	NA	NA
Tab Card	No	No	Yes	No	NA	NA
Bar-To-Hollerith Conversion Unit	A&M 9620	NA	NA	Not Needed	Not Needed	Not Needed
Dial-To-Hollerith Conversion Unit	NA	Not Needed	NA	Not Needed	Not Needed	Not Needed
Simultaneous Bar/Mark/Hollerith Reader	NO	NO	IBM 2956	Not Needed	Not Needed	Not Needed
Hollerith-Compatible Card Reader	NA	HP 2761A Optical Mark Reader	IBM Card Reader	IBM Card Rd HP 2761A OMR	Not Needed	Not Needed

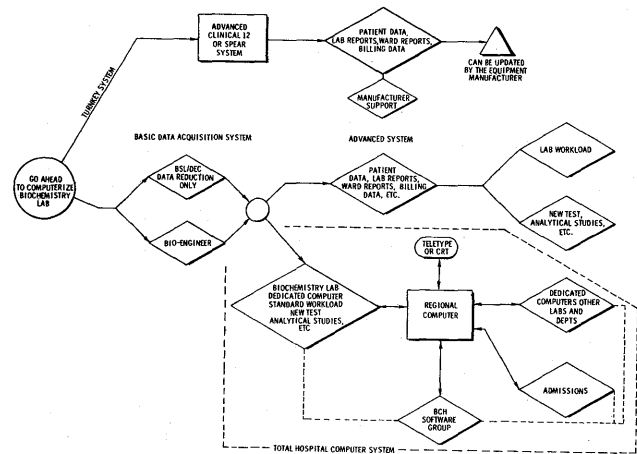


Figure 14—Logic diagram for implementation of a computer system of BCH (1970)



Figure 15—Basic data acquisition system

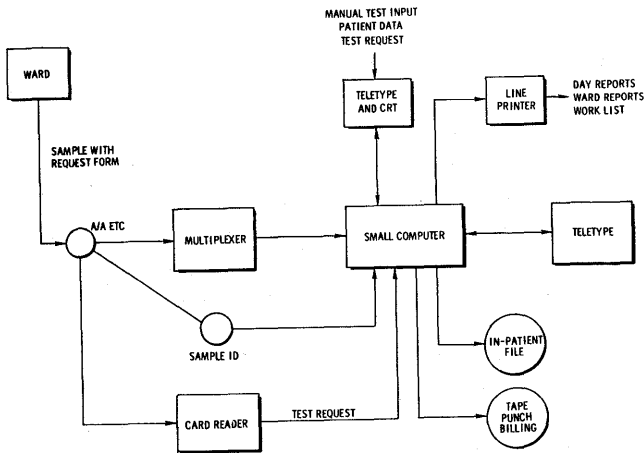


Figure 16—Advanced computer system

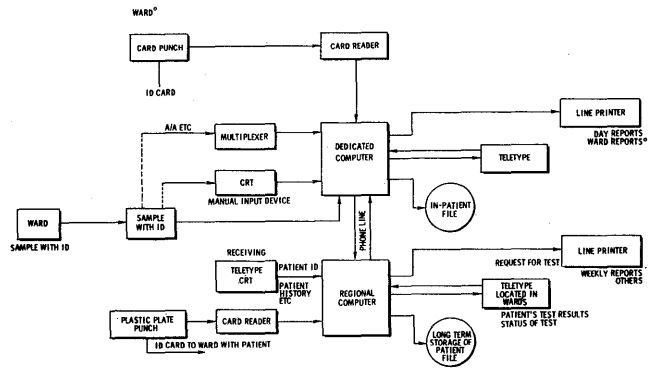


Figure 17—Integrated computer system

who says that his savings did not come from relieving technicians from their mounting clerical duties but from the use of the computer as a lab management system.

If maximum involvement is what the lab desires, they should proceed in the following manner.

Negotiate with a contractor such as D.E.C.; IBM; or B.S.L., for a simple data acquisition system which will do the peak detecting from the auto-analyzers and print out the results and prescribed measurements on a teletype. An example of this type of system is shown in Figure 15. This would eliminate some of the technicians' clerical work and reduce the mounting workload.

The next phase would be to hire a Bio-Engineer who will familiarize himself with lab operations and require-

	Present Lab Operation		Data Acquisition System		Dedicated Computer System		Integrated Computer System	
	No. Technicians	No. Hours	No. Technicians	No. Hours	No. Technicians	No. Hours	No. Technicians	No. Hours
Worklist Preparation	2	1.5	2	1.5	0	0	0	0
Autoanalyzer Tests	10	5.5	5	5.5	5	5.5	5	5.5
Semiautomated Tests	1	7.5	1	7.5	1*	7.5*	1*	7.5*
Manual Tests	1	4	1	4	1*	4*	1*	2*
Results Entered on Worklist	12	3.5	12	2.5	0	0	0	0
Patient File Sorting	3	2.5	3	2.5	0	0	0	0
Result Card Completion	12	5	12	5	0	0	0	0
Result Card Check	1	3	1	3	0	0	0	0
Result Card Corelation	3	2	3	2	0	0	0	0
Result Card Refiling	3	2.5	3	2.5	0	0	0	0
Result Card Ward Distribution	1	.75	1	.75	1	.75	0	0
Maintenance	N/A							
Telephone Call	3	4	3	4	1	4	1	2
TTY Inputs	N/A	N/A	.2	.5	2	2.5	2	.2
Computer	N/A	N/A	1	4	1	8	1	8
	215 Technicihn Hours		173 Technician Hours		56 Technician Hours		46 Techniciean Hours	

* More time available for new tests

TABLE IX

Analyst-Hour Savings by Different Computer Configurations

TABLE X

Cost Analysis of Computer Configurations * 1

Acquisition System	Time Shared System	In-Lab Dedicated Computer
DEC (Basic)	Meditech	DEC Advanced Spear
BSL Chem Lab		BSL (Chem Lab)
IBM System 7		DNA (ClinLab)
2000	Include in Cost	\$8000
50,000	75,000	200,000
52,000	150,000	208,000
54,000	225,000	216,000
56,000	300,000	224,000
58,000	375,000	232,000
68,000	750,000	312,000

Prices are averages for the different groups.

Equipment can be leased from a commercial leasing company at a cost of approximately \$34.00 per thousand or a 36 month lease, or \$22.00 per thousand on a 60 month lease.

ments. The computer system can now be expanded by adding computer memory and expanding the software to include patient file. This would enable the lab to produce the required reports, ward reports, cumulative reports and billing data.

Expansion can be accomplished in several ways. One way is for the lab to have a stand-alone system. If this is what the lab prefers, the contractor can expand the basic lab system to an advanced system. (Figure 16.) An alternative approach which we prefer would be more complex but give better service and be more flexible, is to have a regional computer station set up whereby the other labs in the hospital as well as other departments could use a dedicated computer in their area with direct tie-in to the larger computer or regional computer (Figure 17). This would eliminate the duplication of patient files and programming. It would also aid in that there would be a core of software experts who could help the individual labs to use their facility more efficiently. These people should be from the in-house EDP personnel. An advantage of this would be a large data bank from which all types of statistical studies

could be made. As more users were connected to the regional computer the cost would be spread out among the different departments and their budgets would be lowered.

If enough users were available it would become practical to have teletypes of CRT type terminals at all the nurses' stations, and also at admissions and convenient places throughout the hospital. At present it would not be practical for any lab to support such large satellite-type terminals for just one particular type of lab data. With everyone using it, it would become very practical.

From Table IX it is seen that with any configuration the savings in technician salaries will more than offset the cost of equipment within three years. Table 9 was drawn up assuming 100 percent efficiency which cannot be achieved immediately and the excess in technician capacity could be utilized for running more chemical tests.

Table X represents different financial ways of computerizing the operation and indicates cost over a ten year period for different configurations.

Integrated information system

by J. C. PENDLETON

McDonnell Douglas Astronautics Company
Huntington Beach, Calif

INTRODUCTION

With computer hardware, we hear various computers described as second generation, third generation, etc. This means that the technology used to build these computers has developed through several distinct stages. Generally, each new stage has brought with it order of magnitude increases in price-performance.

Computer-based information system technology has gone through similar stages. However, while current hardware is in the third generation and is moving into the fourth, information system technology appears to be attempting to break out of the second generation.¹ As a result, most current computer-based information systems are not meeting users' needs, particularly timely response and the availability of coordinated outputs.^{2,3,4}

It is suggested that the users' needs can be met by means of Integrated Systems and Data Bases.^{5,6} These terms are used with a great deal of imprecision and misunderstanding. "Integrated Systems" is particularly troublesome in this respect since there are several aspects of Management Information Systems that can be integrated. In addition, these aspects can be integrated separately or in combination. Therefore, this paper first defines and describes these terms, discusses the various types of integration and relates integration to data bases. Then it discusses the rationale for Integrated Information Systems and Data Bases and describes the hardware-software architecture needed to support an Integrated System with a Data Base.

ALTERNATIVE INFORMATION STRUCTURES

There are a number of alternative philosophies for structuring computer-based information systems. These alternative structures stretch from completely Integrated Information Systems to completely Independent

Information Systems and including all combinations in between.

Independent information systems

As the name implies, Independent Information Systems are independent of each other. In the pure form, no information would pass directly from one to another. If the output from one became the input to another, the user would hand transcribe the data to be keypunched. Each system would have all input being keypunched and all output printed.

Independent Information Systems lead to the conventional data base architecture. The term "data base architecture," as used here, describes the files which make up the data base, the way they are organized, the way they are accessed for maintenance and retrieval, including the hardware and software needed to perform these functions. Currently, the typical data base is made up of a large number of files which are stored on magnetic tape. Generally, each file is independent of the other files. Each file is accessed for maintenance and retrieval by its own set of programs. Frequently, maintenance and retrieval are performed on the same run. Generally the outputs of each system tie back to the inputs. Typical systems are shown in Figure 1. The examples shown represent the different ways that data can enter or leave the computer system.

In an environment in which each information system is independent of the others, the data contained in the data base is generally available only to the "owner" of the information system. In addition, there is a certain amount of overlap among the various information systems. This increases the cost of processing, increases the cost of storage, and may cause inconsistencies in the data. Frequently, the reports produced by the various systems are not directly useful. Rather, the reports form a "data base" from which useful information is manually derived by the user.

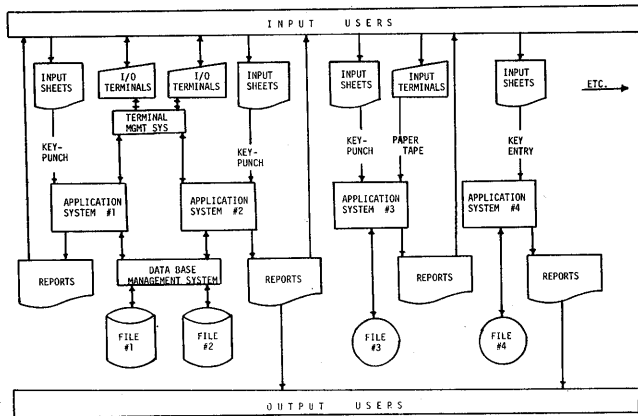


Figure 1—Typical current architecture

Integrated information systems

At the other extreme is the Integrated Information System.^{7,8} However, when the concept of Integrated Information Systems is analyzed, it is apparent that there are several functions that can be integrated. These functions can be integrated in varying degrees and in addition, combinations of functions can be integrated. As a result, there are an almost endless list of different varieties of integration. In order to simplify the situation for analysis and evaluation, the number of alternatives has been reduced to nine. These nine types of integration are discussed below.

1. Integrate the Data Into Data Bases

This means that instead of the data residing in a number of unrelated files, the data are stored in planned fashion in order to provide retrievals. The exact fashion the data are stored in depends on the results to be achieved. The data can be stored to reduce redundancy, facilitate maintenance, expedite access, reduce storage costs, etc.

The result of integrating the data into data bases is that the outputs available to the user (retrievals) are integrated. This means that the user can get all of the information he needs about a task, event, etc., as a unit. He does not have to get one piece of information from one place and another piece from another place. Data for the user is arranged so that all of the information he needs to know about a particular topic is presented as a unit no matter where it originated. For example, three items of concern on most aerospace contracts are cost, schedule, and technical performance. Cost, schedule, and

performance can be considered independently from each other. For example, the actuals for each can be compared with the corresponding estimates or budgets. However, this doesn't tell the whole story. Each factor must also be considered in light of the other factors. Being overspent and ahead of schedule might be good, but being overspent and behind schedule is something else. Thus, it is desirable to integrate the various pieces of information about a particular topic so as to form a more complete story.

2. Integrate Data Processing Functions

In this model, applications are no longer individual computer programs. The functions performed by these programs are now performed by a group of functionally-oriented modules. For example, data capture is handled by a common input processor, regardless of the source of the data.

Another function handled in common for all systems is data management. The data management system works with and maintains two different sets of files. One set of files contains the actual data in the data base. The other set of files contains information about the data bases such as definitions of the data elements as well as information about the structure of the data base.

Other functions which can be handled in common are inquiry-display, output generation, and terminal management.

Implementation of this type of integration requires an investment in software before any application can be implemented because most of the applications will use many of the same basic software components.

3. Integrate Data Flows

In most companies, there is a natural flow of information. In a manufacturing company, for example, the start is in engineering. Then, the mainstream flow of product information is to manufacturing planning, manufacturing, and testing. Financial information would have a different natural flow. When we speak of integrating the data flows, this means to divide the company's system of information flows into modules in a planned way so that the outputs of a processing module can be used directly as the inputs to other modules. In addition, all data which is needed downstream in the processing is collected at the source.

4. Integrate the Data into Data Bases and also Integrate Data Processing functions. (1 and 2 Combined)

TABLE I—Evaluation of integration alternatives

IMPLEMENTATION ALTERNATIVES	IMP COST	OP COST	USER COST	IMP SCH.	OP SCH.	USER RESPONSE	OUTPUT COMPLETE-NESS	ACCURACY & INTEG. OF OUTPUT	AMOUNT OF PLANNING REQUIRED	WEIGHTED TOTAL
1. Integrate D.B. Only	4-20	4-28	7-29	4-20	6-18	8-80	5-50	6-30	4-20	315
2. Integrate DP Functions Only	4-20	4-28	1-7	3-15	6-18	2-20	3-30	6-30	4-20	188
3. Integrate Data Flows Only	4-20	4-28	2-14	4-20	5-15	2-20	7-70	6-30	4-20	237
4. Integrate DB and DP Functions	2-10	2-14	8-56	2-10	9-27	9-90	6-60	8-40	2-10	327
5. Integrate DB and Data Flows	2-10	2-14	9-63	2-10	7-21	9-90	9-90	8-40	2-10	348
6. Integrate DP Functions and Data Flows	2-10	2-14	4-28	2-10	7-21	3-30	8-80	8-40	2-10	243
7. Integrate D.B., DP Functions and Data Flows	0-0	0-0	10-70	0-0	10-30	10-100	10-100	10-50	0-0	350
8. Simulate Integration with Integrated Outputs	8-40	8-56	6-48	8-40	0-0	7-70	0-0	0-0	8-40	288
9. Continue same as present	10-50	10-70	0-0	10-50	1-3	0-0	0-0	0-0	10-50	223
Weighting Factor	5	7	7	5	3	10	10	5	5	

NOTE: The left-most number is the value on a scale of 0-10. The right-most number is the weighted value. Ten is the most desirable (unweighted) evaluation.

5. Integrate the Data into Data Bases and also integrate Data Flows.
6. Integrate Data Processing functions and also integrate Data Flows.
7. Integrate the Data into Data Bases, integrate the Data Processing functions, and integrate Data Flows.
8. Integrate the Outputs
This alternative simulates an integrated data base from the users' standpoint, by giving him the same retrieval capacity he would likely have in an integrated data base. This can be done by using integrating networks, by copying files, or by using retrieval programs like Informatics Mark IV.
9. Perform No Integration (Continue with Independent Systems)

Evaluation of alternatives

Not all of these alternatives are equally desirable from a cost-benefit standpoint. There are a number of ways to evaluate these alternatives. For convenience, a simple numeric rating system is used here as shown in Table I. The left column of the table shows the nine

alternatives. The headings of the remaining columns represent the criteria by which the alternatives are being evaluated. Each implementation alternative is evaluated on a scale of 1-10 for each criteria. The value 10 is the most desirable. Then, each value is weighted with the most important criteria receiving the largest weight. The weighting factors are shown in the last row. The last column on the right represents the weighted total for each alternative.

Based on this evaluation, alternatives 1, 4, 5, and 7 appear to be the most attractive. Alternative 8 is also attractive because it could serve as a bridge to help move from Independent Systems to Integrated Systems. Since alternative 7 represents the ultimate in integration, the following discussions of integration are based on that alternative.

COMPARISON OF ALTERNATIVES

Independent information systems

Advantages

1. Each system is designed and implemented independently.

2. Failure of one system generally doesn't affect the others.
3. Only simple programming, software and hardware technology are required.
4. New systems can easily be added without affecting other systems.
5. Security, back-up, and recovery are easily handled.

Disadvantages

1. Response for non-standard outputs is generally inadequate.
2. Reports requiring coordinated data are difficult to produce.
3. Extracting corresponding data from different files is generally difficult. The needs of the users are not always being adequately fulfilled because the data they require is in separate, independent files.
4. Outputs of systems tend to form data bases from which useful information is manually extracted.
5. The systems tend to be inflexible with respect to producing new outputs.
6. Outputs of one system may have to be manually fed in as input to another.
7. The same data may be stored in several places (and one place may be inconsistent with the other place).
8. Inconsistencies may be introduced because editing requirements and definitions may vary from system to system.
9. The exact meaning of data is often ambiguous because data with the same name in different systems is not identical.

Integrated information systems

Advantages

1. Integrated systems can provide service to users with a response appropriate to the requirement.
2. Integrated systems can provide more useful service to users and coordinated reports become technically feasible.
3. Integrated systems reduce data errors and inconsistencies caused by having the same information in two or more places. Data ambiguities are reduced.
4. Integrated systems allow data retrievals to be uncoupled from data acquisition and file maintenance. This permits changing the systems with

- minimum impact on the user and changing one part of a system without impacting other parts.
5. Integrated Information Systems save money and are more cost effective considering:
 - a. The cost saving in user clerical processing.
 - b. The value of having the needed information on time.
 - c. The cost saving resulting from having correct, nonambiguous and consistent data.
6. Makes inquiry systems feasible, which in turn, increases the availability of data.
7. When coupled with an inquiry system, Integrated Systems allow the user to stop using the computer printout as a data base. With an inquiry system, the user does not have to print out voluminous reports to protect himself because he *may* need to know.
8. Integrated systems provide an opportunity to install extensive checking on data entry.
9. Manual transcription of data for reentry can be eliminated.
10. New outputs can readily be produced.

Disadvantages

1. The design of Integrated Systems is more difficult because the system must be designed as a whole unit rather than as a collection of independent parts. In addition, correction of design errors tends to be more difficult.
2. Since there is less data redundancy, and since there may be considerable coupling between different sets of data, the propagation of undetected errors can cause a lot of damage to the data base.
3. A higher level of hardware and software technology is required for Integrated Systems. As a result, hardware and software for Integrated Systems costs more than the hardware and software for Independent Systems.
4. Security, back-up, and recovery can be problems.

Comparison summary

Independent Systems are generally simpler and, as a result, are easier to implement and operate. However, they don't always meet the real needs of the user. Integrated systems generally have an involved concept and are difficult to implement. However, the payoff is that they can provide a more useful and responsive service to users. It is difficult to make accurate and meaningful generalizations about the comparative cost of the two systems. However, it is probably safe to say

that a really useful and responsive service to users can be provided less expensively by an integrated system.

THE TOP MANAGEMENT VIEW

Thus far, we have seen the case for Integrated Information Systems from the working levels of a company. There is also a case to be made for Integrated Systems from top management point of view. From this standpoint, the central question regarding integrated systems is not which is the best way to store and process data, but rather what is the best way to manage the company? The information system should reflect the way the company is to be managed. If the chief executive wishes to manage the company as a number of separate entities, then Independent Information Systems are to be preferred. If he wishes to manage the company as an integrated unit, then the information systems should be organized as an integrated unit. If certain parts of the company are best managed as independent units and others are best managed as integrated units, then a mixed strategy is to be preferred.

To help identify the role of information systems in a company, let's review the way that companies develop. At the beginning of the Industrial Revolution, businesses were run by individual entrepreneurs who did the planning, selling, producing, accounting, and other necessary jobs.⁹ The basic data and information needed to carry out these duties were filed in the entrepreneur's head. His data processing system was integrated because there was a central source of information from which sprung all policies, plans, and decisions. However, the tasks began to grow more complex and he couldn't handle all of them himself. He hired people to help him and then set up functional responsibilities, such as accounting, production, engineering, and assigned people to these various functions. But as business evolved, the entrepreneur found he could not train people himself. Colleges and universities offered to assist in this task. As university curricula were developed, fences began to appear which were to become the functional boundaries within a business—the accounting profession was formed, the advertising profession, etc.

This is not to say that these functional boundaries were logical in nature, it just happened to be the way they were recognized within a business organization. With the entrepreneur, we had a natural integrated system, but the growth of functional boundaries determines the kind of data utilized by the various groups and tends to limit the scope of the problems and the vision of the individuals tackling these problems.

It is for this reason that we find the computer helping with engineering problems, manufacturing problems, accounting problems, etc., and not necessarily with the company problems. The head of each organizational unit is concerned with his own parochial interests which tend to take precedence over the broader company view.

This suggests a real challenge for management information systems designers. The information systems must somehow provide the various functional heads, as well as top management, the information they need to coordinate their activities to achieve the same effectiveness as the single entrepreneur. The integrated systems concept is a prime vehicle for accomplishing this—emphasizing that while the whole is made up of parts, the parts must not be of such a specialized and unique character that their function as part of the whole is forgotten.

There have been benefits from computerizing individual application areas and this will continue to be the case. However, the real benefits and payoff from computerization will come from systems which recognize the interconnection of subsystems, the common base from which they draw their input, and the meaningful and integrated management reports which form the "action" output of the system.

Thus, from this standpoint, we see that an integrated information and control system is a management-oriented system conceived and designed as a single, total entity to operate and control an entire organization. It does not evolve as a result of the development of many more or less independent applications. In an integrated system, the individual applications must be designed to meet the needs of a restricted area of the organization, but with the whole organization in mind. This results in a system welded together by data flows where unneeded redundancy in data storage and the transmission of useless information from one area to another are eliminated.

AN INTEGRATED SYSTEM ARCHITECTURE

In a large manufacturing company, such as an aerospace company, there is a great diversity of activities and functions. Many of these functions are tied closely together. However, other groupings of functions and activities appear to be rather loosely connected from an information systems standpoint. This is not because these groups are really independent, but rather the links are so complex and nebulous that an integrated system which includes them is beyond present system technology.

As a result, there is a need for a mixed strategy in

which the main activities of the company are integrated, in addition to providing for independent systems. With this in mind, a system architecture is designed which provides for both Integrated and Independent Systems.

To do this, some new terminology is required and this will be discussed first.

Mainstream data

The main purpose of keeping data in an on-line data base is to make it available for inquiry. Some data files are logically interconnected while other data files are logically independent. The data files that are closely interconnected are called *mainstream files*. The other files are called *independent files*. The mainstream files reside in the *mainstream data base*.

Generally, mainstream data is closely associated with the cost, schedule, and performance aspects of the design, manufacture, and test of end-items to be delivered to customers.

Data entering the computer for the first time is called *source data*. Labor hours, as entered on a time card, are source data. A table relating budget function to work-in-process element, shop order, and department is source data. A table relating shop order to contract is source data. Source data can also be defined as data entering the computer system on which no prior logical nor algebraic process had been performed. Sometimes, data is received from a vendor which, by its nature, would not be considered source data in our own shop. However, by convention, this type of data will be included in the collective term "source data".

Mainstream source data is that source data which is required to generate and maintain the mainstream files. *Mainstream systems* are those groups of computer programs which generate and maintain mainstream files. Generally, mainstream systems are connected together by data flows between the mainstream files. A mainstream system reads information from the mainstream data base, processes it in some fashion, and places the results back into the mainstream data base.

Independent files are maintained by *independent systems*. Data required by the independent systems may come from the mainstream data base or may come in as source data. Independent systems *do not* alter any data in the mainstream data base. Independent files may be queried through terminals, and data in independent files can be related to the data in the mainstream data base only through the independent system.

The mainstream data base

The development of an Integrated Information System requires that basic information be gathered from key source documents and used throughout the system. It is also important to develop basic master files which are used in common by the various subsystems. These common master files are referred to as the Data Base.

The Data Base holds all relevant information about a company's operation in one readily accessible group of files. These files are arranged so that duplication and redundancy are minimized. Information concerning on-going activities is captured once, validated, and entered into the Data Base. Normally, the Data Base is subdivided into the major information subsets which are needed to run a business. Each subset corresponds to an Integrated Information System. The key element in the Data Base concept is that all parts of a subsystem utilize the same Data Base in satisfying their information needs.

The Mainstream Data Base has the following characteristics:

1. Data redundancy is minimized; this
 - a. reduces processing cost,
 - b. reduces storage cost,
 - c. reduces data inconsistencies, and
 - d. reduces misunderstandings about the meaning of data.
2. Data organization is unified in the sense that all information about an entity (person, end-item,

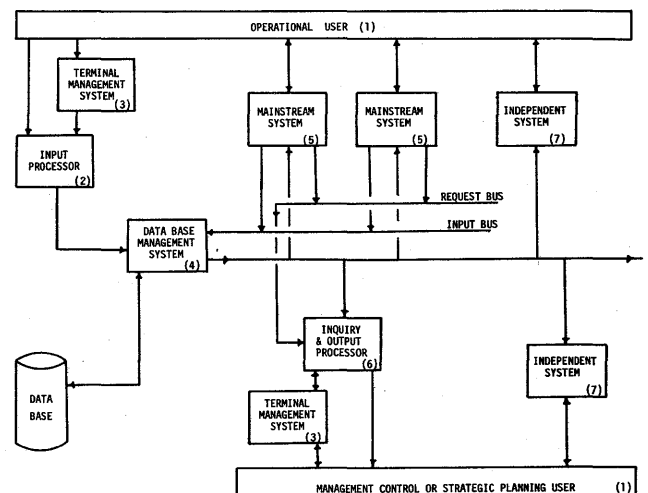


Figure 2—Integrated system architecture summary

- event, department) is available to anyone with a "need to know".
3. Data is available on a timely basis.
 4. Data is accurate.
 5. Inquiries and responses can be handled by a terminal.
 6. All the mainstream data is contained in the data base. However, the data base may be physically divided into sub-data bases.

Pictorial representation

In the Integrated System Architecture Summary shown in Figure 2, the files have been integrated into a group of functionally-oriented data bases on direct access storage devices.¹⁰ In this model, most applications are no longer individual computer programs. The functions performed by these programs are now performed by a group of functionally-oriented modules which are described below. The paragraph numbers correspond to the numbers in the boxes of Figure 2.

1. Users. The users are not part of the computer system, but they interface with it. As a result, they are shown here to provide a more complete picture.
2. Input Processor. Data captured for the data base is handled by a common input subsystem, regardless of the source of the data. A number of different ways of getting source data into machine-readable form are available. After the source data is entered into the data base, it is then available for inspection by the user and also available for processing by the various mainstream systems. The data which results from the mainstream systems are placed into the data base where they are available to the user.
3. Terminal Management System (TMS). All communication between terminals and the remainder of the system is handled through a single interface called the Terminal Management System.
4. Data Base Management System (DBMS). All communication with the Data Base is through a single data base management system. There may, however, be several data paths between the DBMS and the data base. The DBMS works with and maintains two different sets of files. One set contains the actual data in the data base. The other set contains information about the data base, such as definitions of the data elements and information about the structure of the data.

5. Mainstream Systems. The processing of data is handled by individual modules in much the same fashion as at present. Data is obtained from the Output Bus. The results are returned to the Data Base on the Input Bus. "Bus" is used here in the sense of a connector through which any data can be received or dispensed.
6. Inquiry and Output Processor. Requests for reports are received either directly from the user or from a mainstream system. After compilation, the reports are sent back over the terminal network or by the batch processes, as appropriate.
7. Independent Systems. As explained previously, not all of the users' requirements dictate the use of a central data base. The non-integrated requirements are satisfied by Independent Systems.

Functions of the modules

The previous section described the general functions of the module. This section goes into more detail. The numbers in the boxes in Figure 3 correspond to the numbered comments below:

1. Users

Different groups of users have different relationships to the computer system. In this model, they are grouped into two types: "Input" users and "Output" users.⁴ The Input user can also be called the "Operational" user. The Output user is a composite of the "Management Control" user and the "Strategic Planning" user.¹¹

 - a. Input Users. The Input users supply the mainstream source data to the data base. They also get back housekeeping reports describing data errors, file accuracy, processing problems, etc. Input users are responsible for insuring that the Data Base is complete, accurate, and current.
 - b. Output Users. The Output users are the people for whom the data base is maintained. They can make inquiries to the data base by means of a terminal or batch request. Responses to their inquiries can be returned to the terminal or through the batch output generator.
2. The Input Subsystem
 - a. The Input forms are prepared as a result of the natural process of doing work. They are not especially prepared for the purpose of data base input. Both new master records and

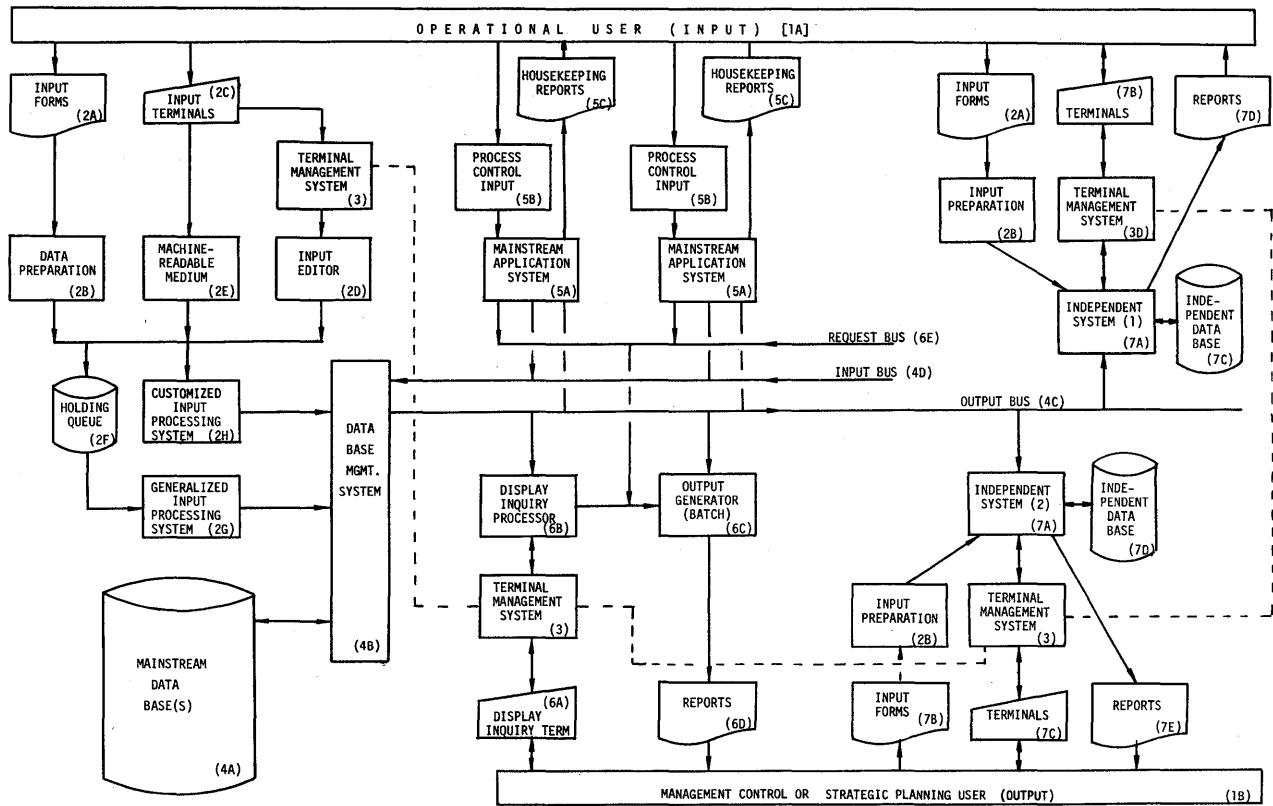


Figure 3—Integrated system architecture detail

- information about events (transactions) are entered through this subsystem.
- b. Data preparation includes keypunch, key stations, OCR, or other central facility transcribing devices.
- c. In addition to the input forms, source data may be captured by means of terminals. These can be two-way terminals like the IBM 2740 or 2260 or can be one-way terminals like badge readers. With the two-way terminals, the user can receive feedback regarding the correctness of the data entered. This is not possible with the one-way terminal except in a very limited sense.
- d. The function of the Input Editor is to prevent erroneous data from entering the data base and to provide feedback to the Input User regarding the accuracy of the data being entered. For example, if the user tries to enter alphabetic data in a field defined as numeric, he would receive an error message. The Input Editor should reside in the same computer as the Terminal Management System.
- e. Not all terminals are necessarily on-line to a

- computer. Some of them may transmit data to a central site where it is written on magnetic tape or other machine-readable medium.
- f. Not all of the data received by the Input Subsystem is necessarily entered immediately into the Data Base. It may be stored in a holding queue for batching or other purpose.
- g. The Generalized Input Processing System takes each piece of source data, associates it with the appropriate identifications, and stores it in the data base.
- h. Not all of the input for all applications can be handled by the Generalized Input Processing Subsystem. Input for a particular application may, if desired, be processed by a Customized Input Processing System when the general system is inadequate.
- 3. Terminal Management Subsystem (TMS)
 - The TMS handles all communications between the terminals and the rest of the system; e.g., polling, receiving, transmitting, etc. The TMS block is shown at several spots on the chart. These boxes are connected with a dotted line to indicate that they are all part of the same

TMS. The TMS can be implemented as a stand-alone computer or as a part of a larger computer.

4. Data Storage and Management Subsystem

- a. The purpose of the Mainstream Data Base is to provide a repository for the mainstream data. However, additional housekeeping data is also stored with the mainstream data. The housekeeping data consists of the names of the data elements whose values are stored in the mainstream data base, the locations where the values are stored as well as information about the data storage organization. For convenience in maintenance and usage, the Mainstream Data Base can be logically divided into sub-data bases.
- b. The Data Base Management System is the channel by which data is entered into or is fetched from the data base. The DBMS translates the logical READs and WRITEs into physical addresses and causes the data to be transmitted and checked.
- c. All subsystems are considered to communicate with the DBMS, and hence, the Data Base, by means of the Input and Output Busses. Any information in the Data Base is available on the Output Bus (provided access restrictions are met).
- d. Derived information, as created by an Application Process, enters the Data Base from the Input Bus. No subsystem other than the Mainstream Systems can put information on the Input Bus.

5. Mainstream Systems

Mainstream systems are computer programs or groups of computer programs that generate derived data from source data and/or other derived data. Most of the existing computer programs would become mainstream systems in this architecture. Mainstream systems are not normally permanently resident in the main computer memory. They are kept in a library and called when needed. There is no limit to the number of Mainstream Systems.

- a. The actual processing is carried out in a processing module.
- b. In addition to the source and derived data (master records and events) from the Data Base, a number of table or other control inputs may be required; e.g., Federal, State, and Local tax tables.
- c. Generally, the processing results in one or

more housekeeping reports. These reports describe the accuracy and completeness of the processing.

6. The Inquiry and Display Processor Subsystem

- a. The Display and Inquiry Terminals are the devices used by the user to retrieve data from the data base. Output resulting from requests can either be displayed on the terminal or sent out over the batch processing system. Physically, these terminals may be the same as the input terminals, but they don't necessarily have to be.
- b. The Display and Inquiry Processor interprets the requests received from the terminals and causes the request to be satisfied either by a display on the terminal or by transmission to the Output Generator for batch processing.
- c. The Output Generator receives requests for reports either from the Display and Inquiry processor or from a mainstream system. The Output Generator selects the required data from the Data Base, resequences the data if necessary, summarizes and prepares the reports.
- d. The reports produced by the Output Generator generally go back to the Output User.
- e. Requests for reports are placed on the Request Bus by the Mainstream Application Systems.

7. Independent Systems

Independent Systems are inherently simpler and less costly than Integrated Systems when the power of an Integrated System is not required. Thus, the architecture contains Independent Systems to meet those less stringent requirements. While only two Independent Systems are shown, there can be any number.

Independent Systems are similar to Mainstream Systems in the sense that they are specific application-oriented. However, they differ in several respects as indicated in the explanation of the following modules.

- a. Independent Systems can use information from the main data base, but cannot put information into the main data base. The Independent System processor module generally executes in the batch mode, although it could operate in the time-sharing mode.
- b. Independent Systems may have their own input which does not become part of the source data in the main data base.

- c. Input can also be entered by means of terminals, if desired. These terminals can be physically the same terminals as used elsewhere in the system, but they don't have to be.
- d. Independent Systems can create and maintain their own exclusive data bases. The file can be resident on disk, drum, tape, etc.
- e. The Independent System reports can go back to the Input or to the Output User.

REFERENCES

<p>1 RICHARD G CANNING <i>What is the status of MIS</i> EDP Analyzer Vol 7 No 10 Oct 1969</p> <p>2 RUSSELL L ACKOFF <i>Management misinformation systems</i> Management Science Vol 14 No 4 December 1967</p> <p>3 NORBERT L ENRICK <i>Why management information systems fail</i> ASTME Vectors 6th Issue 1969</p>	<p>4 T WILLIAM OLLE <i>MIS: data bases</i> Datamation Vol 16 No 15 Nov 15 1970</p> <p>5 JOHN DEARDEN <i>How to organize information systems</i> Harvard Business Review Mar-Apr 1965</p> <p>6 PAUL COLLINS <i>The utilization of real-time management information systems in the aerospace industry</i> Presented to Institute of Management Sciences Los Angeles 20 Oct 1970</p> <p>7 ROBERT G MURDICK <i>MIS development procedures</i> Journal of Systems Management Dec 1970</p> <p>8 WILLIAM M ZANI <i>Blueprint for MIS</i> Harvard Business Review Vol 48 No 6 Nov-Dec 1970</p> <p>9 JEROME KANTER <i>The computer and the executive</i> Prentice Hall 1967</p> <p>10 RICHARD G CANNING <i>Trends in data management part 2</i> EDP Analyzer Vol 9 No 6 June 1971</p> <p>11 ROBERT N ANTHONY <i>Planning and control systems</i> Harvard University Press 1965</p>
---	---

A machine independent fortran data management software system for scientific and engineering applications

by IAN HIRSCHSOHN

Integrated Software Systems Corporation
San Diego, California

INTRODUCTION

Increasing interest in management information systems had led to much activity in developing techniques and software for data management.^{1,2,3} Virtually all of the attention has been focused on the specific needs of commerce. Elaborate data structures have been developed for accessing the same information via many different attributes^{4,5,2} and considerable attention has been given to the acquisition of data from multiple terminals operated by unskilled clerks.⁶

In spite of this flurry of activity, little attention appears to have been given to the specific data management needs of the scientific and engineering communities. The development of automated data acquisition instruments interfaced to magnetic tape recorders or on-line computer channels has led to a data explosion in many fields. There is a real need to be able to sift, catalogue and store this data with a minimum of tears and a maximum of efficiency. Unlike commerce, scientific data does not usually have multiple attributes and is characterized more by its sheer volume and non-standard format. Thus an elaborate data structure is usually superfluous and flexibility in handling is paramount.

Most of the data management packages presently in existence were developed specifically for commerce and are highly inflexible in handling non-standard data manipulations, which renders most of them almost useless to the scientist and engineer.

This paper will describe the details of a software system called DISSCUS (Disk Integrated Software System for the Control of Utility Storage) written by the author specifically for the management of scientific and engineering data. DISSCUS^{7,8} is fully operational on the UNIVAC 1230/490 system at the Naval Undersea Center in San Diego. It has been designed to be as

machine and hardware independent as possible and has been written as a system of interlinked FORTRAN IV subroutines.

The author does not propose to present any new concepts in data management, he simply seeks to coach the relevant existing concepts in terms of the neglected needs of the scientist and engineer.

This paper is oriented toward the poorly informed scientific programmer rather than the knowledgeable EDP professional. The terminology is specifically defined and existing jargon is deliberately avoided in order to focus on the problem rather than differences of definition. The only claims made by the author for the system is that it is simple, flexible and works.

REQUIREMENTS FOR SCIENTIFIC AND ENGINEERING DATA MANAGEMENT

Before discussing any details of the software, it is instructive to formulate the basic data management requirements of the scientific and engineering communities. The author will attempt later in this paper to try to correlate the features of DISSCUS with those requirements. In order to provide some basis for proposing the requirements, they will be derived from four diverse problems whose needs are common to a vast range of problem areas.

Linear programming

This involves processing many large matrices, usually of different sizes, which cannot all fit into core simultaneously. They are frequently too large to fit into core as a unit and must then be partitioned. Similar difficulties are often experienced in problems involving linear algebra, factor analysis or analysis of variance.

For this type of problem the data management system should possess the ability to:

- (i) transfer blocks of data of different length to and from bulk storage at random,
- (ii) identify the blocks through suitable indices, preferably alphanumeric names,
- (iii) group blocks of data (corresponding to matrix partitions) under a single index and be able to operate on both individual blocks and entire groups.

Contour plotting⁹

In most contour plotting applications involving empirical data, the surface interpolation technique is arbitrary and in many cases the user may be aware of a "bump," discontinuity or slope condition of which the mathematics is not.¹⁰ For elaborate contour charts the computation time is usually large and recomputing the entire plot just to change the layout can be expensive. Thus it is desirable to be able to edit parts of a contour, delete and add contours, and store entire charts in such a way that they can be retrieved or edited later. These needs are common to the whole class of data-base graphics, e.g., topography, cartography and charting physical properties. To be useful for these problems the data management system should, in addition, possess the ability to:

- (iv) transfer a file of data blocks from bulk storage to tape and restore the file upon request,
- (v) delete or insert data blocks,
- (vi) retrieve data blocks sequentially (for plotting contours).

Text editing

In order to be able to manipulate the text of manuscripts, papers or manuals it is desirable to be able to insert, add, delete or replace individual words, sentences, paragraphs, subsections and whole chapters. It is also desirable to be able to keep copies on tape and retrieve the text sequentially for printing. Deletion of large quantities of text, at random, ultimately leads to a large accumulation of discarded fragments that could cause overflow of the bulk storage limits. It is therefore desirable to be able to compact the non-redundant text. Considering paragraphs or subsections as data blocks, the data management system should also have

the ability to:

- (vii) "garbage collect," i.e., remove discarded data blocks,
- (viii) add, delete or insert segments within a block.

Time series analysis

The statistical analysis of seismic recordings, electroencephalograms, electrocardiograms and anemometer recordings by computer presents singular problems in manipulating data records on existing magnetic tapes produced by laboratory or field recorders.¹¹ Bad records are often present due to equipment malfunction or operator error and frequently the data requires complex unpacking, scaling and calibration.¹² The needs of this problem are common to a rapidly increasing community of users utilizing automated digitized data acquisition from a wide variety of measuring instruments. The data management system should also possess the ability to:

- (ix) assemble a contiguous file of data records from arbitrary run sequences of records and/or files on a user's tape,
- (x) index records or store them *without* extraneous headers (should the user wish to use his own reading routines),
- (xi) form a library of data files on a single tape with facility for cataloguing the content and nature of each file,
- (xii) massage user records prior to transfer, with a standard action, e.g., real to integer conversion, or return the record to the user for manipulation should he so desire.

CONSIDERATIONS IN DESIGNING PRACTICAL AND REALISTIC SOFTWARE

The worth of software is best measured, not by the sophistication of its instructions, nor by the elegance of its algorithms, but by the satisfaction of its users. It is far too common to find software that is sophisticated, elegant and impractical. It appears to be the vogue to concentrate on the computer sciences aspects of the software rather than its ease and effectiveness in solving a diversity of real problems. A package that is simple to use, clearly documented and easily interfaced to other software is far more useful than a slick, context-free language that attempts to be all things to all users. If the latter provides little facility for interfacing

other software its use will often be limited to undergraduate instruction in computer sciences.

The guidelines used in the development of DISSCUS were that it should:

- (1) be capable of interfacing a user's FORTRAN program with a minimum of effort,
- (2) be as machine independent and flexible as possible,
- (3) incorporate as much error diagnostic facility as possible and spare no error messages,
- (4) provide facility to return control to the user, should he so desire, wherever feasible,
- (5) require minimal user knowledge of the operating system,
- (6) not require any special jobs to be run,
- (7) have instructions that can be easily summarized,
- (8) assume as much as possible if not told otherwise by the user,
- (9) use a minimum of parameters, with mnemonics wherever possible,
- (10) avoid jargon in its documentation and specifically define all terms.

Based on these "ten commandments" the usual approach of writing an interpreter with instructions read from cards was rejected because it violates (1) and (6) totally. Arguments advanced for this approach are that the user needs no familiarity with the host language and that an instruction syntax can be devised that is more compatible with the application as in the case of LISP, SNOBOL and LISTAR. In practice, however, violation of (6) is a nuisance and (1) will preclude its use for a vast range of applications. Furthermore it is the author's experience that it is relatively straightforward for even a mediocre programmer to write a customized program to interpret cards and make the necessary subroutine or precompiler calls.

A pre-compiler has flexibility of instruction format, but is highly dependent on the operating system and also violates (2) and (6).

The remaining approach is that of a procedure or subroutine package. FORTRAN was selected as opposed to ALGOL or PL/1 because although the latter are prettier languages, the real world tends to speak FORTRAN in spite of all its idiosyncracies.

This approach usually leads to long parameter strings to cover all the options, in violation of (9). In order to avoid this, the author configured DISSCUS instructions as a system of subroutine and function modules which are linked through several labelled COMMON blocks.¹⁰ As further instructions are specified, their parameters are transmitted through the

COMMON blocks. This technique divides the parameters among several routines and avoids redundant specification. One problem with this technique is that as the user's program becomes more sophisticated, the number of subroutine calls increases to a point at which they are difficult to follow.¹⁰ As a compromise, the author wrote a string interpreter that could interpret a Hollerith string into component instructions. Thus the instruction modifiers could be specified as a single string, for example:**

```
CALL TPEFIL('SOURCE=MYTAPE(UNIT=2,
KEY=1234)*NAME OF FILE=FILE5*
RECORDS=2-30, 55, 60-80*FILES=1-6*WORDS/
RECORD=500*MODE=REAL$')
```

This technique allows the instructions to be specified in a format that is easily understood by the user and reaps many of the benefits of an interpreter and pre-compiler without violating any of the basic guidelines. It does, however, make it difficult to modify the parameters during program execution. To avoid this DISSCUS permits alternate specification of the parameters as an array of triples, so that the above example could also be specified as:

```

DIMENSION INSTR(42)
      :
Triple 1 { INSTR(1) = 'SOUR'      Operator
          { INSTR(2) = 'MYTAPE'  Argument
          { INSTR(3) = ','       Separator
Triple 2 { INSTR(4) = 'UNIT'     Operator
          { INSTR(5) = 2         Argument
          { INSTR(6) = ','       Separator
Triple 3 { INSTR(7) = 'KEY'
          { INSTR(8) = 1234
          { INSTR(9) = '*'
      :
          { INSTR(13) = 'RECO'
          { INSTR(14) = 2
          { INSTR(15) = '-'
          { INSTR(16) = 'RECO'
          { INSTR(17) = 30
          { INSTR(18) = ','
          { INSTR(19) = 'RECO'
          { INSTR(20) = 55
          { INSTR(21) = ','
      :
INSTR(42) = '$'
CALL TPEFIL(INSTR)      (B)
```

** All interleaving blanks and the characters) / and . are ignored.

The instructions are considered as triples having the form:

Operator = Argument

Only the first three characters of the operator phrase are interpreted; the rest are ignored.⁷ The triples may be interspersed in any order provided that the separator is not a (.

This alternate technique is more cumbersome even if the DATA statement is used, but allows the user to compose or modify his parameters with ease. Another method of specification is to use the FORTRAN statement—ENCODE.⁷ Unfortunately many FORTRAN compilers do not recognize the ENCODE statement so that the triples method was devised as an alternative in order to maintain machine independence.

The argument strings are not discussed in detail in this paper; they are used in examples where necessary.

ASSUMPTIONS MADE ABOUT THE COMPUTER SYSTEM

In designing DISCUSS it was assumed that a system on which it could be implemented has the following minimal features:

- (a) a FORTRAN IV compiler which implements the full specifications of USASI FORTRAN IV,
- (b) a bulk storage random access device such as a disk, drum or bulk core,
- (c) three, or more, magnetic tape drives preferably capable of transferring a buffer asynchronously with computation.*

These minimal features are to be found on most medium and large scale computers and even a large number of small scale computers. No assumptions were made regarding the optimum length of disk or drum sectors. These lengths are set internally and the software system can be reoriented to a given random access device with ease. DISSCUS accesses and positions the disk or drum and the tapes through a hierarchy of internal positioning and read/write routines. The latter are written in such a way that if the system can provide sophisticated software for these functions, it can be incorporated relatively easily. The author assumed a minimum system tape package namely—read/write a buffer of given length or position forward/back by files/records and return status.

Many larger installations have provision in their

operating systems for ascertaining parameters such as record length or byte size, so that some of the DISSCUS specifications may be unnecessary in these cases. The author has found through bitter experience, that although tailoring software to a sophisticated system may be expedient, the software will die with the system and be assumed as little as possible.

BASIC ORGANIZATION OF DISSCUS

The fundamental unit of DISSCUS is a *data segment*; this is defined to be a block of data having arbitrary length and content that can fit into core as a complete unit. A data segment could be a matrix, a matrix partition, a contour line (or a part of it), a subsection of a manuscript or an interval of a time series.

A collection of segments is termed a *file*. A file could be a map, a contour plot, a manuscript or a time series.

Files are assembled in a *data silo*. Thus a single tape is usually used for a data silo, but a silo could also be kept on disk or drum.

DISSCUS is organized into three distinct, but connected, sections so that the features of one can be used without familiarity with the others.

Segment management is concerned with the storage and retrieval of data segments within a specified file.

File management is concerned with the transfer of files to and from a data silo.

Data manipulation is used to operate on data within a given segment, for example, compressing and decompressing the segment treated as string of characters, or building list structures within it.

SEGMENT ADDRESSING

If an index is assigned to each segment as required by (ii) above, keeping track of the indices presents a problem, particularly in contour plotting and text editing in which the number of segments may run into thousands. It is therefore unfeasible to maintain a single table of indices in core. If the table is kept in bulk storage, an efficient look up scheme is desirable due to the high temporal cost of bulk storage references. One efficient technique that is frequently used is hash addressing¹³ whereby an item is located in a table by computing an address based on the index value. In the class of problems considered by the author, the natural grouping of segments provides the key to a more efficient scheme, namely referencing each segment by two indices—the name of the group and its name within the group as illustrated in Figure 1.

This scheme has the disadvantage that two indices must be used in referencing a segment, but in practice

* Bulk disk can be used instead of tapes.

it is more natural to think of a segment as part of a contour, chapter, matrix or other group. This scheme does have several additional benefits—firstly segments in different groups can have the same name without ambiguity, secondly entire groups can be manipulated naturally as units, e.g., entire contours, chapters or matrices may be retrieved or deleted in a single operation. It does not preclude elaborate data structures such as that of LISTAR⁴ because these can be developed as trees or rings of pointers contained within one, or more, segments pointing to others.

Problems such as time series analysis and contour plotting are naturally oriented toward sequential storage and retrieval. For these problems advantage can be taken of the ability of many devices, such as tapes, to store or retrieve asynchronously with computation and thereby decrease the job time significantly. This led the author to provide two modes of segment management—*random* and *sequential*.

In random segment management the segments may be operated upon at random regardless of their order of storage, whereas in sequential management, the segments must be processed in order of storage. If, for example, the groups in Figure 1 were stored in order from left to right, it is permissible to operate on segments—A(2), 300FT(CURVE2), 1(1), QQ(G), A(1A) and 1(2) in that order only in random mode; if the mode is sequential the order must be A(1A), A(2), QQ(G), 1(1), 1(2), 300FT(CURVE2).

A random segment file is usually processed by a bulk storage device, whereas a sequential file is usually processed through tape. If the sequential operations only involve retrieval or the creation of a new file, just one tape is necessary, otherwise both a current file tape and an updated file tape are necessary.

SEGMENT OPERATIONS

Most segment operations are performed by a set of FORTRAN functions. The reason for using functions rather than subroutines for most of the operations, is that a function provides a cheap and ready method for returning an error condition or other information to the user.

Define *quantity* to be either a group or a segment and *list* to be the respective existing file or group.

The basic operators that are available (together with their mnemonics) are:

- Add ADD* Adds the quantity to the end of the list
- Delete DRP* Deletes the quantity from the list
- Insert INS* Inserts the quantity after a specified quantity on the list

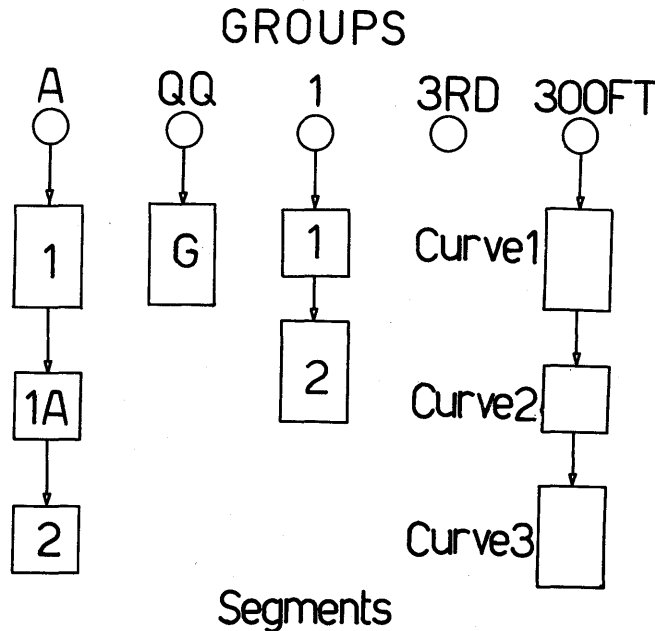


Figure 1—Segment addressing scheme

- Retrieve GET* Transfers the quantity to a buffer area or array supplied by the user
- Replace RPL* Deletes the quantity and replaces it by another, i.e., a combined DRP and INS
- Edit EDT* Overwrites an existing quantity by another with *no deletion*
- Stack BEG* Adds the quantity at the beginning of the list
- Retrieve Next NXT* Transfers the next quantity on the list, following the last GET or NXT, to a given buffer.

The quantity is denoted by another mnemonic as follows:

- NDE random group
- LST random segment
- NSQ sequential group
- LSQ sequential segment.

Combining the quantities mnemonic with that of the operator yields the function name for the desired operation.

Table I summarizes the standard functions and their arguments; use of this technique is best illustrated by the examples that follow.

Example 1. A linear programmer wishes to store two partitions PART1 and PART2 each having 1000

TABLE I—Standard Segment Management Functions and Arguments

Suffix	Prefix	Random by		Sequential by	
		Group NDE	Segment LST	Group NSQ	Segment LSQ
Add	ADD	a	f	a	f
Delete	DRP	a	e	a	e
Insert	INS	b	g	b	g
Retrieve	GET	d	f	d	f
Replace	RPL	b	g	b	g
Edit	EDT	-	f	-	f
Stack	BEG	a	f	a	f
Retrieve Next	NXT	c	c	c	c

Argument Index	Number Arguments	Parameter List
a	1	(DELETED or NEW GROUP)
b	2	(NEW GROUP, PRECEDING or REPLACED GROUP)
c	2	(BUFFER ARRAY, ARRAY LENGTH)
d	3	(GROUP, BUFFER, LENGTH)
e	2	(GROUP, SEGMENT)
f	4	(GROUP, SEGMENT, BUFFER, LENGTH)
g	5	(GROUP, NEW SEGMENT, PRECEDING or REPLACED SEGMENT, BUFFER, LENGTH)

words, later retrieve them as a unit in the work space IWORK, operate on them and then overwrite the existing partitions with the new partitions. The instruction sequence might be:

```

      :
IDUMMY = NDEADD('DIST')
IDUMMY = LSTADD('DIST', '1ST',
PART1, 1000)
IDUMMY = LSTADD('DIST', 2, PART2,
1000)
IDUMMY = NDEGET('DIST', IWORK,
2000)
      :
Operate on IWORK
      :
IF(LSTEDT('DIST', '1ST', IWORK,
1000).LT.O) GO TO 100
IF(LSTEDT('DIST', 2, IWORK(1001),
1000).GE.O) GO TO 200
100 PRINT 101
101 FORMAT('ERROR IN DISSCUS OP.')
STOP
200 CONTINUE
      :

```

NDEADD defines a new group DIST and the two calls to LSTADD tack the partitions onto DIST. NDEGET retrieves the group as a unit and LSTEDT edits the existing partitions. The first 4 calls do not make use of the function argument whereas the last two will terminate the job if an error occurs. Unless told otherwise (see below), DISSCUS will print any errors and proceed. The mode of the index names is unimportant as long as they are single computer words.

Example 2. The user of Example 1 now wishes to insert the partition PART1 before the segment 1ST and PART2 after 1ST. He then wishes to transfer the group, segment by segment, to form a new group—TIMES.

```

      :
I = LSTBEG('DIST', 0, PART1, 1000)
I = LSTINS('DIST', '1A', '1ST', PART2,
1000)
I = NDEADD('TIMES')
I = LSTGET('DIST', 0, IWORK, 1000)
J = 1
300 I = LSTADD('TIMES', J, IWORK, 1000)
I = LSTNXT(IWORK, 1000)
J = J + 1
IF(I.GE.O) GO TO 300
      :

```

This example illustrates the use of sequential retrieval by LSTNXT without having to bother with index names; LSTGET is necessary, however, to start the ball rolling. Note the use of the error condition (negative value) for testing for the end of the operation.

Use of the sequential mode, activated by the prefixes NSQ and LSQ, follows the same pattern as their counterparts NDE and LST. In this mode, however, the files are generally on tape and it is necessary to specify either the source or destination silo, or both, depending on whether a new file is created, an existing file is read or a file is an update to an existing file. BEGSQ (see Table II) is used to initiate operations on a sequential file and ENDSQ terminates operations on the file as illustrated by:

Example 3. A time series analyst wishes to drop segment TIME2 and then insert a 250 word segment from the array BLOCK after the segment TIME5 in the series with group having the name HEART. He then wishes to create a new series LUNGS after HEART, composed of 50 segments of 400 words each which he will read himself. The current file with name JONES is on silo WARD3 and the updated file is to be placed on WARD5. Finally he wishes to retrieve and operate on the 20 segments following TIME5.

```

CALL BEGSQ('SOURCE SILO = WARD3*
  DESTINATION = WARD5*NAME OF
  FILE = JONES$')
I = LSQDRP('HEART', 'TIME2')
I = LSQINS('HEART', 'TIME6', 'TIME5',
  BLOCK, 250)
I = NSQINS('LUNGS', 'HEART')
DO 100 J = 1, 50
  :
  READ 400 words into IWORK
  :
100 I = LSQADD('LUNGS', J, IWORK, 400)
  CALL ENDSQ
  CALL BEGSQ('NAME = JONES*
  SOURCE = WARD5$')
  DO 200 J = 1, 20
  IF(J.EQ.1) I = LSQGET('HEART',
    'TIME5', IWORK, 400)
  IF(J.GT.1) I = LSQNX(T(IWORK, 400)
  :
  Process segment in IWORK
  :
200 CONTINUE
  CALL ENDSQ

```

TABLE II—Additional Subroutines and Functions
for Segment Management

(see References 7 and 8 for argument lists)

Name	Purpose
NDEMOR(c)*	Retrieves another portion of a large random group
NSQMOR(c)*	Retrieves another portion of a large sequential group
NDEADG(e)*	Adds a group with garbage as first segment
NDEING	Inserts a group with garbage as first segment
RSTGRB	Discards current garbage chain and starts anew
BEGSQ	Initiates sequential operations
ENDSQ	Terminates sequential operations on current file
PRTSQ	Prints a map of segment names for a file
CLRDSK	Clears the current disk file
PRTDSK	Prints a map of segment names for current disk file
DMPDSK	Transfers the current disk file to a specified silo
GETDSK	Transfers a disk file from a silo to disk
ERROFF	Suppresses printing of error messages
ERRPRT	Resets ERROFF to print errors
ERRSTP	Causes job termination if an error occurs
SQ TO RA	Converts a specified sequential file to current disk file
RA TO SQ	Converts a current disk file to a sequential file

* See Table 1 for argument lists.

The retrieval operation could not be done immediately following the update because HEART lies before LUNGS and this would violate the rule of sequence; this necessitates terminating the update with ENDSQ and then initiating a new sequence using BEGSQ. The argument strings for BEGSQ are self-explanatory and silos are discussed further in the File Management section below.

The contents of segments deleted from the current random file (henceforth referred to as the *disk file*) by NDED RP, LSTDRP, NDERPL or LSTRPL are added onto the end of a "garbage chain." The current garbage chain can be turned into the first segment of new group by NDEADG or NDEING (see Table II). This facility can be very useful for many applications, for example:

Example 4. To edit a manuscript held in the disk file a user wishes to form a new chapter (group), CHAP8, out of the contents of chapter, CHAP2, and paragraphs PARA3 and PARA2 of chapter, CHAP7. The instruction sequence could be:

```

I = NDED RP('CHAP2')
I = LSTDRP('CHAP7', 'PARA3')
I = LSTDRP('CHAP7', 'PARA2')
I = NDEADG('CHAP8', 'PARA1')

```

The garbage chain becomes a single segment with name PARA1.

Working through the garbage chain rather than a retrieve-store-drop-insert sequence is both simpler and more efficient.

Frequently, in using NDEGET or NSQGET, the sum of the contents of the segments of a group may be too large to fit into the workspace available. In this case the rest of the group can be retrieved one portion at a time by using NDEMOR or NSQMOR (see Table II) respectively until the group is exhausted.

An image of the contents of the disk file can be transferred to a silo by using DMPDSK (see Table II). Alternatively a disk file image can be restored from a silo using GETDSK, e.g.

```

CALL DMPDSK('DEST. = TAPE5*NAME =
  PLOT5$')
CALL GETDSK('SOURCE = TAPE2*NAME =
  PLOT3$')

```

writes the current disk file contents onto silo TAPE5 as the file PLOT5 and sets the disk file to the image held by file PLOT3 of silo TAPE2. This facility is powerful for drawing elaborate plots, for example in drawing a contour plot on a map with political boundaries it is not necessary to hold the entire mess as a single

file—the coastlines could be one file, the political boundaries a second and the contour plot a third; furthermore all the files need not be on the same silo.

When many operations are performed on the same file it is easy to lose track of the groups and segments in the file. PRTDSK and PRTSQ print a map of the groups and segments in the disk file or a sequential file respectively.

The routines SQ TO RA and RA TO SQ convert a sequential file to the current disk file or vice versa, respectively. This facility provides a simple means for garbage collection as demanded by requirement (vii) above, e.g., the sequence

```
CALL RA TO SQ('DEST=A TAPE*NAME=
DUMMY$')
CALL SQ TO RA('NAME=DUMMY*SOURCE=
A TAPE$')
```

causes the disk file to be written onto the silo, A TAPE, with all superfluous fragments removed by RA TO SQ. SQ TO RA writes the same file back onto the disk minus all garbage.

The other subroutines in Table II are self-evident and will not be elaborated further.

DATA SILO FUNDAMENTALS

Maintaining silos on tape requires designing the software system around the physical and practical limitations of tape. The first anomaly of tape is that if a data block, or record, is written into the middle of a tape file, generally the next record is unreadable even if the update has the same length as the original. Second, the time to position the tape to a specified record and file is usually long. Third, tapes are extremely prone to parity errors. There is also the practical consideration that the data contained on the silo may be valuable and if an error occurs during the update, the original data should be preserved at all costs.

In order to alleviate these criteria, DISSCUS has two types of silo—a *transactions* silo and a *master* silo. Each silo is identified by a header record at the beginning of the tape, containing the type and name of the silo; in addition the user may assign his own key to the silo. Before any operation is performed on the silo, DISSCUS checks to see whether the tape is a valid silo and that the name and key agree with those specified. Writing on a master silo is strictly forbidden except when an update is made and then only on the updated version, so that if the update should fail the original is still intact. Transactions silos are used for writing disk

file images and sequential segment files; a file may only be added at the end of the last file written. There is no restriction on reading from a silo provided that it passes the validity checks. Files may be transferred from a transactions to a master silo as described in the next section.

The actual tape records of a silo file usually have uniform length to simplify reading. Thus sequential file segments may be broken across several records or suitably blocked, but this is transparent to the user.

Each file on the silo has a unique name and the silo maintains a directory of the names of its files as well as a catalogue containing a summary of the file contents, e.g., the type of file, number of records, record length, date and job number of making. The catalogue of a master silo is more extensive than a transactions silo and has facility for storing and editing user remarks, as well as other basic information such as the number of bytes/word, byte length, date and job number of last update, original source tape and mode of storage. DISSCUS carefully checks the directory and catalogue of a silo before reading from it, to see whether the file exists and the operation is valid. The author deliberately made DISSCUS as detail conscious as possible, because he has yet to find a user who kept a completely current notebook of the exact contents of all his tapes. Time and again the author has seen users waste hours of computer time analyzing the wrong data and he is one of the worst offenders.

Finally, in order to satisfy requirement (x), the directory and catalogue for a master silo are held separate from the data files.

TABLE III—Summary of File Management Subroutines
(see References 7 and 8 for arguments)

Name	Purpose
NEWSLO	Turns a tape into a silo
SETSLO	Initiates a silo update
XECSLO	Executes the silo update
PRTSLO	Prints a map of an existing silo
TPEFIL	Creates a contiguous file from a list of records and files on a specified user tape
ADDRCS	Adds or splices a list of records and files from a user tape into an existing file
XFRFIL	Transfers a file from another silo to the update
DRPFIL	Deletes a file from the update
REMARK	Adds, deletes or inserts a remark line on the silo catalogue
PRTOFF	Suppresses printing of summaries
SETFIL	Initiates retrieval from a user made file
LSGNXT	Retrieves an arbitrary segment from a user file
LRCNXT	Retrieves a record from a user file

FILE MANAGEMENT OPERATIONS

All file management operations must be conducted on a legitimate silo. An arbitrary tape can be converted into a silo by NEWSLO (see Table III), e.g.,

```
CALL NEWSLO('TRANS. SILO=
TAPE4(KEY=C5GA)')
```

creates a transactions silo with name TAPE 4 and assigns it the key C5GA. Once the silo has been created it is available for operations such as disk file image transfer or sequential segment operations. Frequently it is desirable to transfer important files from a transactions silo to a master silo, this is done by performing an update, e.g.,

Example 5. The user wishes to transfer the file JONES from silo WARD5, PLOT5 from silo TAPE5 and PLOT3 from silo TAPE2 onto a new master silo SILO1:

```
CALL NEWSLO('MASTER=SILO1(KEY=
1234)')
CALL SETSLO('NEW SILO=SILO1(KEY=
1234)')
CALL XFRFIL('SOURCE=WARD5*
NAME=JONES')
CALL XFRFIL('NAME=PLOT5*SOURCE=
TAPE5')
CALL XFRFIL('NAME=PLOT3*SOURCE=
TAPE2*INSERT AFTER FILE=JONES')
CALL XECSLO(O, O, O)
```

NEWSLO turns SILO1 into a master silo with key 1234. SETSLO initiates the update and defines the participating silos. If the silo was assigned a key, this must be specified in parentheses after the silo name regardless of the operation. The update instructions (XFRFIL in this case) are stored as an internal disk file and executed by XECSLO. Note that the physical position of a file can be specified by using the INSERT instruction as illustrated in the third call to XFRFIL.

Building up the update instructions as a disk file and then executing them has several advantages over performing the instruction as it is encountered. Firstly, the user does not have to follow a rigid sequence in specifying the instruction subroutines and this may be vital in some problems in which the instruction information is not available at the time at which the instruction would otherwise have to be called. Secondly, if there is any error in an instruction no tape operation takes place, saving an incredible amount of time in practice. Thirdly, the directory and catalogue can be fully updated before execution and can therefore be placed

conveniently at the head of the master silo. Finally, if the tape controller is capable, the internal tape positioning operations can be optimized to look ahead on the list of instructions and position idle tapes ahead of time, resulting in enormous time savings.

One of the most useful features of the file management section is its facility for handling users' data tapes as required by (ix) and (xii), e.g.,

Example 6. A user wishes to add several files derived from his tapes MYTAPE and RUN5, to SILO1 of Example 5 as specified by the sequence:

```
CALL NEWSLO('MASTER=COPY1')
CALL SETSLO('OLD SILO=SILO1(KEY=
1234)*NEW SILO=COPY1')
CALL TPEFIL('NAME ASSIGNED=EXPT4*
WORDS/REC=150*BYTES/WORD=8*
BITS/BYTE=4*RECORDS=50*MODE=
TWOS COMPL.*FILES=1, 7, 20-50, 7, 5, 1-10*
SOURCE TAPE=MYTAPE')
CALL TPEFIL('SOURCE=RUN5*RECORDS=
1-500, 600-650*FILES=2-7*DROP
RECORDS=50, 65, 100-150, 605, 620-640*
WORDS=250*NAME=EXPT2*INSERT
AFTER=JONES')
CALL TPEFIL('NAME=CONV1*SOURCE=
MYTAPE*MODE=REAL*ORIGINAL REC
LENG.=150*WORDS=600*RECORDS=
100*ACTION=USER')
CALL REMARK('4TH EXPT', 'NAME=
EXPT4')
CALL REMARK('PACKED STORAGE',
'NAME=EXPT4')
CALL REMARK('UNPACKED STORAGE',
'NAME=CONV1')
CALL ADDRCS('RECORDS=10-35*FILE=9*
ACTION=REAL TO INTEGER*NAME=
EXPT2*SOURCE=RUN5')
CALL REMARK('COMBINED RECORDS',
'NAME=EXPT2')
CALL NEWSLO('MASTER=COPY2(KEY=
AG2)')
CALL XECSLO(IWORK, 1200, 'COPY
SILO=COPY2(KEY=AG2)')
```

NEWSLO turns COPY1 into a master silo with no key and SETSLO initiates the update with SILO1 as the original and COPY1 as the updated silo. The first call to TPEFIL adds a file having name EXPT4 with 150 words per record, containing 8 bytes/word, of 4 bits each in twos complement.* The file is to be transferred from the user's tape MYTAPE and is to consist of the first 50 records from each of the files 1, 7, 20 to 50, 7, 5 and 1 to 10, concatenated in that order. The

second call to TPEFIL illustrates the use of the DROP option, i.e., records 50, 65, 100 to 150, 605 and 620 to 640 will be deleted during the transfer of the record sequences from the files 2 to 7 of the user's tape RUN5; the file will be inserted after the existing file JONES (on SILO1). The third call to TPEFIL illustrates the user interaction facility. In this case records consisting of 150 words each are to be read from the first file of MYTAPE (default FILE is file1). After each record is read DISSCUS calls a *user supplied subroutine with the name USER* having the contents of the current record, the file name, original record number, and original file number as arguments. The user may then perform whatever operations he pleases on the record. Upon return to DISSCUS it is transferred as a 600 word record to the update silo and the fact that each word is a real number is noted.

The calls to REMARK each add a line of user remarks to the silo catalogue for the file named in its argument. ADDRCS adds a further set of records to the file EXPT2 previously initiated and each word of every record is converted from real to integer before it is written onto COPY1.

XECSLO actually performs the instructions which were hitherto stored in the disk file. IWORK is a workspace buffer with at least 1200 words to accommodate two of the longest records to be encountered, viz., 600 words for file CONV1.

During an update, a summary of each instruction is printed to insure that there is no misunderstanding and the directory and catalogue are printed at the conclusion of the update. This printing can be suppressed by PRTOFF and PRTSLO can be used if a map of the contents of an existing silo is desired. The new silo tape is rewound at the end of the update and checked for hardware induced errors.

SETFIL may be used to initiate subsequent retrieval of information from a file constructed by TPEFIL and LSGNXT or LRCNXT are then used to retrieve a single block of information. LSGNXT retrieves the next segment of data of specified length regardless of whether it crosses record boundaries, whereas LRCNXT retrieves the next record. These routines are provided for the user's convenience, but he is free to use his own.

DATA MANIPULATION

This section is still under development. Currently routines exist for compressing and decompressing segments consisting of card image blocks, and for string

* These specifications may be unnecessary on installations on which they can be retrieved from the system.

manipulation. These routines are straightforward and will not be elaborated further.

It is anticipated that at the time this paper is presented, routines will exist for forming segments consisting of tree and ring structures for pointers to other segments. This would be useful for problems requiring elaborate data structures, but would not impose unnecessary overhead or complication on the majority of users who do not need such capability.

NOTES ON THE IMPLEMENTATION OF DISSCUS

DISSCUS combines many techniques described in detail in previous literature^{4,5,14} so that only a brief description of these techniques is given.

For the random mode segment operations, DISSCUS uses a paged virtual memory scheme.¹⁴ Bulk storage, or disk, is divided into N pages, n of which reside in core (see Figure 2). A fresh data chunk of length x is placed in the core page, j , for which the available space $S_j \geq x$ and $(S_j - x) \leq (S_k - x)$ for any $k \leq n$ otherwise a fresh page is used, i.e., the packing is as tight as possible. A clock count is incremented with each virtual memory reference and demand for a page not in core releases the core page whose clock count $T_l < T_k$ for all $k \leq n$ and $k \neq l$.

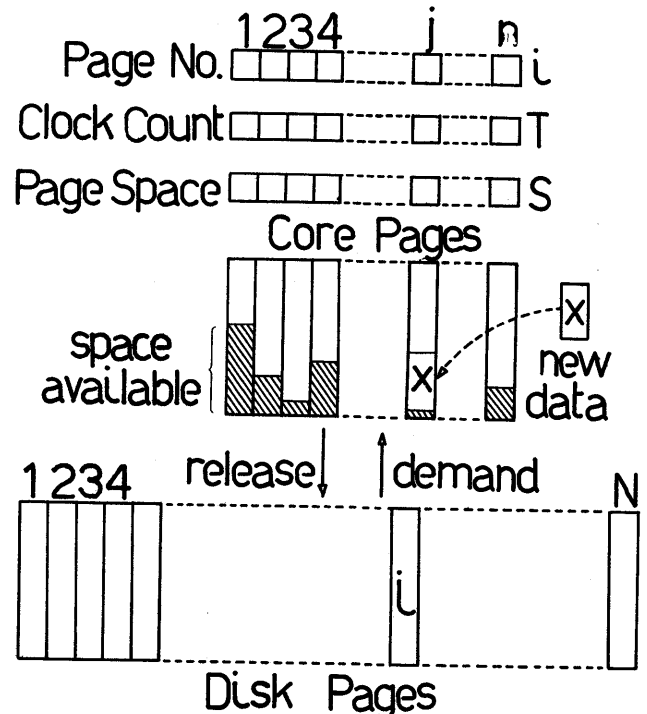


Figure 2—Paging of bulk storage

Large random mode segments are split into chunks across several pages.¹⁴ The segment address table¹⁴ for only one group resides in core at one time. The chunks are linked as a multilist structure⁵ of back pointers as depicted in Figure 3. Back pointers are used rather than forward pointers because this saves referencing the disk to update the forward pointer of the last chunk of the group. Thus the multilist structure parallels an inverted file structure⁵ represented by the segment address table. Forward and backward pointers are kept between the group headers themselves to facilitate insertion and deletion of groups. The signs of the chunk address are used to separate them into segments rather than maintaining extra pointers, in order to save core.

The back pointers are needed for the garbage chain operations and their overhead is negligible. The extensive file management capability encourages the user to break up his data into several different files which often alleviates any need for special garbage collection and compromises the need for expensive disk residency for a large single file.

The sequential segment operations are based on an index sequential scheme⁵ as illustrated in Figure 4. The segments are stored on consecutive records of uniform length and a table of pointers to the first record of each group is kept in core.

The random mode routines are used for storing the silo update instructions. Each file represents a group and its particulars are contained in the first segment.

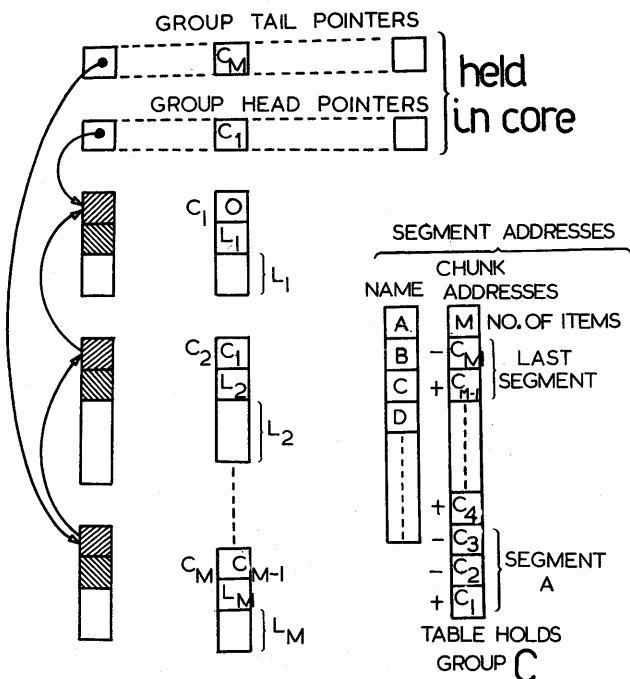


Figure 3—Random segment management

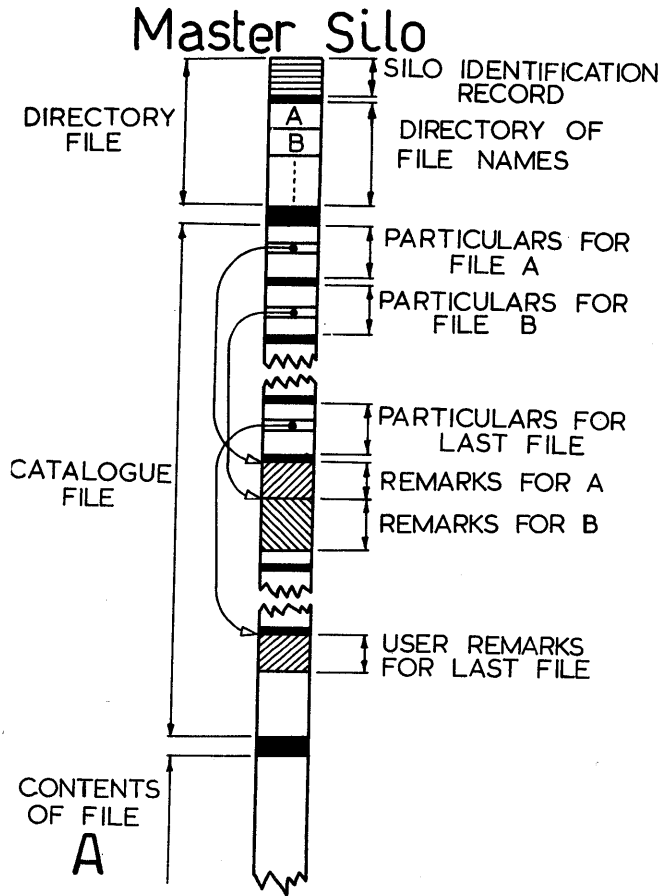


Figure 4—Sequential segment management

Each call to REMARK or ADDRCS adds an update segment to the group. The facility of the random mode operations for inserting, deleting and adding groups or segments is very handy for inserting, deleting or adding file instructions and enables splicing of records into existing files simply by inserting and editing the necessary instruction segments. XECSLO converts the update instructions into a task list of actual tape positioning, reading and writing operations, which is stored as another group. The task list can then be optimized to position idle tape drives if the hardware is capable of it. The task list is then executed to form the updated silo.

Figure 5 shows the composition of a transactions silo. The directory of file names is the last file and is overwritten and restored when a new file is added. It is not kept at the beginning of the tape because it could not be updated when a new file is added.

Figure 6 shows the composition of a master silo. The file particulars are stored as consecutive records of uniform length to facilitate reading and the variable

length user's remarks are blocked at the end of the catalogue file.

CONCLUSION

DISSCUS provides a modular utility software system that is adaptable to most applications. In those areas in which it has limited capability, its design enables the user to utilize as much of it as he sees fit and program whatever else he needs. For example, problems requiring elaborate data structures can be handled adequately by maintaining the trees and rings as separate segments with suitable pointers to the segments containing the actual data. This relieves the user of the chores of paging and segmentation and provides him with the additional capability of file management.

To the casual observer, DISSCUS may not seem to provide any service not currently available in the Job Control Language of some systems such as OS/360. A primary motivation for writing DISSCUS is the extreme difficulty (if at all possible) involved in changing the specifications or sequence of the control instructions

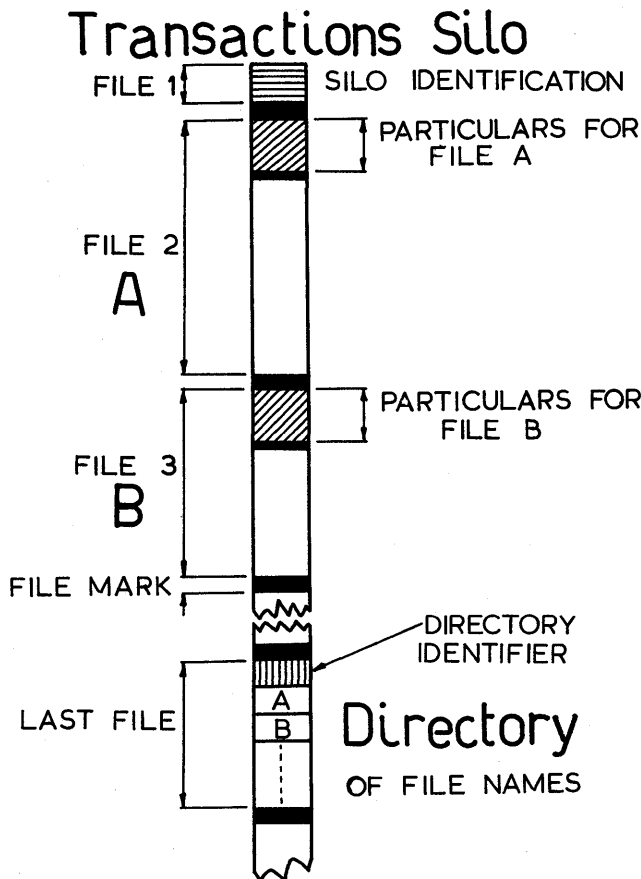


Figure 5—Transactions Silo organization

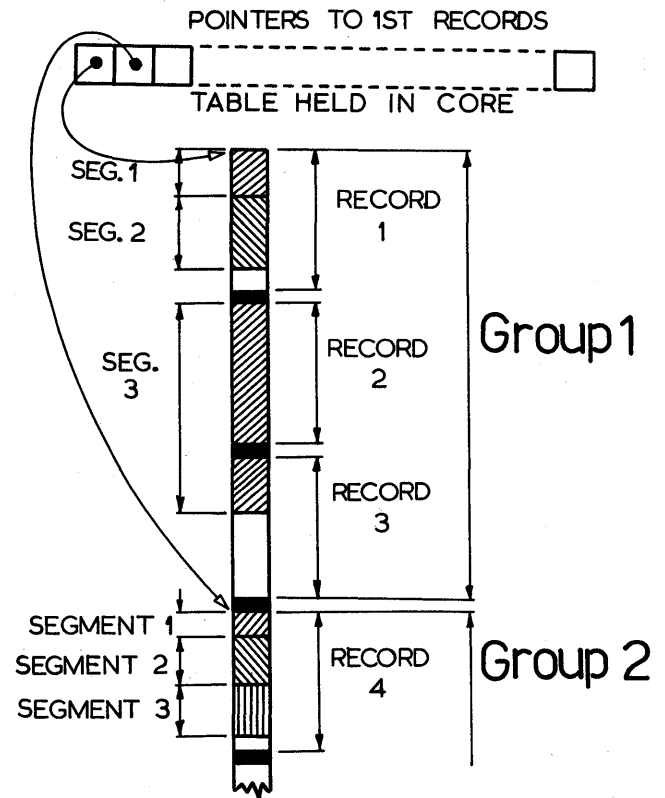


Figure 6—Master Silo organization

during program execution. Dynamic interaction with these instructions is unavoidable in many scientific applications and seemingly versatile Master Control Languages are usually a source of considerable frustration in these areas.

The author hopes that this paper will help stimulate interest in providing practical, user oriented software.

REFERENCES

- 1 A W HOLT
Data structure theory and techniques
Management Information Services
Box 5129 Detroit Michigan 48236
- 2 I FLORES
Data structure and management
Prentice Hall 1970
- 3 P J FRY
Managing data is the key to MIS
Computer Decisions January 1971 pp 6-10
- 4 A ARMENTI et al
LISTAR—Lincoln information storage and associative retrieval system
AFIPS 1970 Spring Joint Computer Conf Vol 36
pp 313-322

-
- 5 G G DODD
Elements of data management systems
ACM Computing Surveys 1 2 June 1969 pp 117-133
- 6 T L CONNORS
ADAM—A generalized data management system
AFIPS 1966 Spring Joint Conf Vol 28 pp 193-203
- 7 *DISSCUS beginner's manual*
Integrated Software Systems Corp
4131 Jewell Street San Diego California 92109
- 8 *DISSCUS intermediate manual*
Integrated Software Systems Corp
4131 Jewell Street San Diego California 92109
- 9 S P MORSE
Computer storage of contour map data
ACM Proc 23rd Nat Conf 1968 (P-38) pp 45-51
- 10 I HIRSCHSOHN
AMESPLOT—A higher level data plotting software system
Comm ACM 13 9 Sept 1970 pp 546-555
- 11 C H GIBSON R R LYON I HIRSCHSOHN
Reaction product fluctuations in a sphere wake
AIAA Journal 8 10 Oct 1970 pp 1859-1865
- 12 I HIRSCHSOHN
TURBSTAT time series analysis subroutines
AMES Department UCSD La Jolla California 92037
- 13 R J BELL C H KAMAN
The Linear quotient hash code
Comm ACM 13 11 Nov 1970 pp 675-677
- 14 P J DENNING
Virtual memory
ACM Computing Surveys 2 3 Sept 1970 pp 151-188

Requirements for a generalized data base management system

by A. C. PATTERSON

Bankers Trust Company
New York, New York

Several approaches to generalized data base management systems have been documented for the benefit of the data processing community. Perhaps the two most significant reports are Codasyl's Data Base Task Group Report issued in October, 1969, and revised in April, 1971, and the joint GUIDE/SHARE Data Base Management System Requirements. Each report brings with it its own distinct jargon to our already over-developed Tower of Babel. Although the reports often advocate similar solutions to the data management problem, there is a clear difference in emphasis and dominant philosophy. Perhaps the most immediately obvious difference between the two reports is that Codasyl has specified an actual COBOL syntax for the implementation of a data base management system (DBMS) and the GUIDE/SHARE report has simply defined the requirements the data processing community would impose on implementors of DBMS's.

The GUIDE/SHARE report is the result of work carried out over a twenty month period beginning in 1969 and culminating in the November, 1970, document. The committee was comprised of representatives from more than forty companies engaged in diverse activities from banking to armed forces, life insurance to machine manufacture, and government agency to university. It is reasonable to assume, then, that the requirements set forth by the committee are indeed representative of the requirements of the data processing community at large.

The GUIDE/SHARE committee was charged to:

Define the users functional requirements for a data base oriented system. The group does not intend to concern itself with implementation methodology at this time.

The document which summarizes the committee's thinking neither pretends nor intends to describe an existing system or one currently in development. The

report is in no way intended as system specification. Some of the requirements may not even be implementable under the current hardware and software technologies. The report is rather intended to state a group of requirements the universe has placed on data base management systems of the future, both near and long range. The committee felt that to limit its thinking to concepts effectively realizable on currently available hardware and within the reasonably primitive software technology of today would be to limit the value of the work by assuring its obsolescence upon publication. Furthermore, it has been the intention to advance the state of the data base art by providing a forum where data base related concepts could germinate and burgeon.

Examine first the functions which are basic to a data base management system. First, there is a **Data Base Descriptive Language (DBDL)** which fulfills among others, the data definition function. The **Data Base Manager (DBM)** is the supervisory function carried out by hardware and software which coordinates all components of the system. The **Data Base Command Language** allows a user to make requests within the Data Base Management System. The **Data Base Administrator** is a function responsible for the definition, organization, protection, and efficiency of the data base(s) of an enterprise. It is a function performed by humans using, among other things, the facilities provided by the Data Base Management System. Hence, we have the components of the system: DBDL, DBM, DBCL and DBA. And, of course, there must be physical data to be manipulated, and that named collection of units of physical data which are related to each other in a specified manner is a data base.

The Data Base Management System has data independence as a primary design criterion. Data will be structured in physical storage in a logical and non-redundant manner, insofar as practical. As new processing requirements imply new structures, dynamic restructuring of the physical stores will be possible

without impacting the current processing. Not only will "access" to data be simple, versatile, and secure, but its rules will be easy to learn and present few problems during its integration with a current system. It will achieve data independence for the user, data reliability, data non-redundancy, data integrity, security, and performance.

Data independence removes the burden of access strategy from the application programmer which frees him to do the job he does best—programming. The programmer should not concern himself with data formats or structures but rather with procedures and data manipulation. How and where the data are stored is to him immaterial. What relationships the data he is using have with other data which he is not using should not be his concern. The DBMS allows the programmer to concentrate on doing his procedural description well.

Data non-redundancy has several virtues which recommend it. Costly storage facilities should not be taxed with the responsibility for keeping duplicate data elements nor should a system be taxed with the maintenance responsibility for redundant data items. Where several different logical record occurrences have an identical data item value, a relationship should be maintained, where feasible, to assure that the materialization of the logical data item always is made from the same physical representation in the physical store. Duplicate data values in the physical store must be minimized or eliminated.

Through control facilities incorporated in the DBMS, data integrity will be provided. There must be a reasonable assurance that data and relationships are maintained accurately. For instance, this requirement includes the necessity for a facility to propagate changes to all derived data elements when a component of the derived data element is changed. If X , Y , and Z are data elements derived according to the following equations:

$$X = A + B$$

$$Y = B * C$$

$$Z = Y + D$$

a change to B (which is a "real" data element as are A , C , and D also) effects a change to X , Y , and Z . These changes must be accurately reflected not only in derived data element values, but also in any associations the derived data elements enjoyed based on their values. A derived data element is a physically stored element derived through some algorithm from one or more data elements.

The DBMS will provide security for data based on installation specification. Security in the system is

essentially bidimensional: sensitivity of data, and authority of user or requestor. The given installation may place security restrictions on individual data items, specific combinations of data items (for instance, one user may be allowed to see salary averages, but only when more than five salaries are part of the computation of the average or he may be allowed to see age without name and vice versa), and of course restrictions may be placed at any higher, more general level of the DB such as the logical file level.

Because many of the ways in which the system performs may be specified by the installation, and because user exits to the DBMS will be provided at appropriate points, the DBA will be able to optimize the system's performance. Consistent, measurable, and tunable performance of a system of this magnitude is a critical design level consideration. It is primarily the performance criteria which obviate the immediate wholesale implementation of the GUIDE/SHARE DBMS.

At this point, it is appropriate to examine in some detail the major components of the system, that is the DBA, DBDL, DBM, and DBCL. The requestor's view of the DBMS is critical. He interfaces directly with two components—one is human (the DBA) and one is that hardware/software combination, the DBM. The requestor first communicates with the DBA to have data definitions entered into the Data Base Directory. The Data Base Directory (or DBD) is a part of the DB itself and contains all information pertaining to accessibility of data. That information includes:

- physical to logical mappings (or physical record descriptions);
- logical to physical mappings (or logical record descriptions);
- relationships and associations among data elements and items;
- security and authorization required at the data level;
- miscellaneous information pertaining to the maintenance of and access to the data.

The DBA uses DBDL to enter data definitions and their associations and relationships. The language is non-procedural. It defines, states, or describes to the DBM how the DBM may operate on data. Once the definitions are entered in the DBD, the Requestor may proceed with his access to the DB.

Then, the Requestor communicates with the DBM using the DBCL. The Data Base Command Language, itself non-procedural, states requests in a form that DBM can interpret. Included as primitive functions in the language are the approximate equivalents in current

data management systems of: (1) Open; (2) Retrieve; (3) Replace; (4) Add; (5) Delete; and (6) Close. A primitive command, once invoked, requires by definition no further intervention by the Requestor until its function has completed. Primitives operate only at the logical data level. Any physical ramifications resulting from the invocation of a DBCL primitive are created by DBM and remain unknown to the Requestor.

Available with each primitive command is a parameter list where the Requestor may supply further qualifications for his request. Among the items he may specify are: (1) a logical file name; (2) logical record name or names; (3) data area name; (4) a communications area where DBM and Requestor may transfer information to each other, such as status of request, exit routine addresses, and the like; (5) self-imposed data access restrictions to be used when DBCL primitives are executed. Where parameters are not specified in a given primitive, reasonable defaults will prevail. For instance, if no file name is provided in the "open", SYSIN may be assumed for an input file and SYSOUT for an output file. System-provided default values may be replaced by particular installations at a time approximately equivalent to system generation (SYSGEN). Depending upon the Requestor's own authority or security clearance to access the DB, the sensitivity or security level of the data he wishes to access, and the defined relationships and associations of this data supported in the DBD, he may access the DB in any number of ways. Before the Requestor may begin accessing the DB, he must alert the DBM that he intends to use the system. This operation in DBMS is roughly equivalent to OPEN in current data management systems. The more information the Requestor provides DBM with at OPEN, the less information he will have to provide with each subsequent command. Also, the more DBM knows at OPEN about the kinds of access the Requestor intends, the greater the efficiency with which DBM can satisfy the requests in subsequent commands. In short, the longer specifications are deferred, the more it costs. The following are two of the ways of accessing data: (1) retrieve a record occurrence of a file based on record identifiers (something like keys in currently implemented data management systems); (2) record qualifiers. Clearly, the more description the Requestor provides to the DBM through DBCL the greater the likelihood of a unique "hit", or at least a reduced population of records which satisfy the retrieval criteria.

A few examples of the kinds of qualifications the Requestor might provide are: (a) record identifiers such as account number or part number; (b) qualifiers such as NEXT (nextness for a given logical file being described in the DBD). Other qualifiers available to the

Requestor are previous, first, last, *nth*, a relationship name, a search procedure name where the procedure has been developed by the given installation, and Boolean expressions using AND, OR, NOT, BETWEEN, EQUAL TO, NOT EQUAL TO, GREATER THAN, and LESS THAN. Qualifications used for record searches may be predefined, named, and stored in the DBD or may be explicit with each command. Deletion, Replacement (that is, update), or Addition may be done only when the record being operated upon can be uniquely identified. That is, a record was retrieved based on certain qualifications, but more than one record in the file satisfies all the selection criteria, the record could not be updated (that is, replaced with modifications) or deleted. Only unique occurrences of logical records can be added to a file. The three categories of data used in almost any data base system, namely, the logical data, the entity, and the physical data, are also present in the GUIDE/SHARE DBMS. The largest unit of physical data is the Data Base. The most general reference to the concept of entity is the word itself. The DBM is that portion of the system (along with the DBA) which understands the entity concept. That is, DBM knows that both physical and logical data are organized around an entity or entity type. The entity is any person, place, thing, or event of interest to the enterprise. An entity type is a so-called classified entity, where an entity has been specifically identified as a particular type, such as employees, departments, vendors, tractors, sales, and so on. The entity construct is any association of entity record types and the entity relationships which connect them. When an entity construct has been named, it loses its general connotation to become an entity record type.

There are three types of or levels where relationships can exist—logical, entity, and physical. The physical and logical relationships are self-explanatory. The entity relationship is defined as a named relationship that exists between two entities of the same or different entity types, such as that between employees and departments. The physical correspondences for the logical file are data set, form extent, and space extent. The data set referred to here is roughly equivalent to the data set in current data management systems, or, more precisely, it is a named collection of stored records of one or more stored record types. A data set, in GUIDE/SHARE terminology, is a collection of one or more space and form extents. A space extent is a unit of contiguous space on some medium. A form extent is that portion of a data set in which all stored record occurrences are of the same stored record type—a physical subset of the physical file.

The logical record is the only record a program ever "sees." It is made up of one or more data items which

are materialized from data elements found in one or more stored records, and are moved from physical to logical arrangements and vice versa, according to descriptive information and attributes defined in one entity record. This information is stored in the DBD along with other information the DBMS needs to access data, and there is therefore no tangible entity record or entity record file. The entity record type refers to a named record, that is, a collection of entity field names that represents the attributes of a particular entity type.

A logical record type is a specific collection of one or more data items. An occurrence of a logical record type is called a logical record.

The stored record type bears the same relationship to its physical counterparts as the logical record type bears to its logical counterparts.

A logical record occurrence is a particular instance of a specific group (one or more) of data items which make up a unique logical record. The stored record occurrence is similarly defined on the physical side.

The data item is a unit of logical data which may be either an elementary or group data item. Effectively, it is the smallest named unit of logical data. A similar definition applies to data element. The entity field is a generic term referring to either an elementary or group entity field. An elementary entity field is a named field referring to an attribute of an entity. The group entity field is simply a named association of entity fields within an entity record type.

An entity field type is a named entity field that represents some particular attribute of an entity type.

The data item type is a specific data item; such as an account balance data item, as the stored data element type is a specific data element. Data item occurrence and data element occurrence are specific representations of the values of units of data. Group data item and elementary data item fall under the generic term, data item, and are self-explanatory.

The last concept to be addressed in some detail is the Data Base Descriptive Language. The entity concept has been explained and the interfaces between and among the DBM, DBA, DBCL, User or Requestor, and physical and logical data. The DBDL discussion should clarify the DBA function and how the DBM knows what to do with data.

DBDL is primarily the instrument of the DBA function. This function uses the language to define to the system the necessary physical, entity, and logical descriptors associated with a Requestor's needs. Although the Requestor may define certain logical data locally and temporarily if those logical data are a subset of existing entity descriptors, only the DBA may describe physical data.

The DBA must describe data, define relationships, define mappings, define security and specify performance measurements. These functions may be fulfilled using the DBDL. DBDL statements are interpreted by the DBM and the derived descriptors are stored in a directory (DBD) which is maintained as part of the Data Base. This, then, is the major supporting element of data independence.

The DBDL provides the DBA with the means to:

- Describe the physical and logical characteristics of the data base.
- Describe the relationships which exist between the various components of the data base.
- Describe the rules by which the DBM resolves the differences between the physical data and the logical data (processed by the application program).
- Describe the rules by which the DBM performs security and integrity checks to prevent unauthorized and destructive access to the data base.
- Subset the DBMS and organize the data base to control costs while optimizing performance.
- Monitor and control the day-to-day operation of the DBMS.

There are some constraints placed on the DBDL. First, it must be an independent language and not an extension of current host languages. It must be an extensible language, or one which will provide the primitive functions which can be combined and expanded to fulfill new requirements as they evolve. DBDL is a descriptive and not a procedural language. The DBDL is, however, capable of identifying procedure written in a procedural language by the user which is to be invoked in particular instances, but DBDL cannot describe the procedure itself.

Appropriate defaults are provided in DBDL but with the facility to restate defaults or to override them at a number of different times in the course of processing the DBMS.

The facility to name units of data on all levels (including field, group, file, data set) and define relationships allows the DBD to contain more than one version of each descriptor and define the conditions under which each is to be used. Alias names are supported as well as names with degrees of qualification, or indexed names. This latter facility is provided to reduce ambiguity.

DBDL will define identifiers for records or so-called key fields. Alternate identifiers are provided for. Data attributes and alternative data attributes with rules for their uses are specified in DBDL.

Among its most important functions, the description and definition of logical data relationships in DBDL ranks high. These relationships may be among any number of levels of data (elementary, group, etc.), any number of levels of entity definition, and any number of units of data. The only relationships which the application programmer must be aware of are the logical ones between a logical record and its data items and between and among logical records.

The DBDL allows the DBA to define membership and association rules for files. It also allows the DBA to specify rules for additions and deletions, security requirements, propagation requirements, and the sequence in which logical records will be added to the file. Propagation requirements may include something like this: a man was in the personnel department and was on the salary committee. To be on the salary committee, he must also be in the personnel department. The man transfers to data processing and his department code is updated. He will automatically be removed from the salary committee if the DBA has specified the general rule via DBDL.

Repeating group and elementary data items (including the zero case) may be described in DBDL and interrelationships between and among the files may be defined. Mappings may be defined which are consistent with the host language.

It is through DBDL that the DBA defines entity fields, record types, constructs, and relationships along with the data and function attributes for each entity level. Real, virtual, and derived fields are described in DBDL.

Entity relationships may be defined in terms of entity fields, record types, constructs and any other appropriate entity relationships. There are a number of other facilities which the DBDL provides, especially regarding mappings, space use, indexes, planned data redundancy, security, integrity, performance and control, and so on. These other facilities are more fully described in the GUIDE/SHARE November, 1970, report.

The time frame envisioned for the implementation of a DBMS which satisfies the requirements in the GUIDE/SHARE report is about five years. However, there are in existence today some systems which satisfy some of the requirements. It is not anticipated that any implementor will fulfill all the requirements in a first release. Rather, a system reflecting all the requirements will begin by satisfying a subset of them and then evolve into the sophisticated system proposed by GUIDE/SHARE. The continually advancing hardware technology will, to a degree, determine how soon some of the proposed features will be made available.

Perhaps the most important idea that the data processing community must digest and accept is that there should be one standard philosophy applied to any DBMS development by any implementor. This philosophy must intelligently address the concepts of data independence, data integrity, data relatability, data non-redundance, as well as others. It remains to bring together the various groups which have worked on DBMS definition and to then develop a single, standard approach to data base—one which can be accepted by the data processing community at large. It will not be an easy task, but it must be done.

BIBLIOGRAPHY

- 1 *Data description and access*
ACM SIGFIDET Workshop Proceedings November 1970
Available through ACM
- 2 R M BALZER
Dataless programming
Conference Proceedings FJCC 1967
- 3 R G CANNING
EDP Analyzer Vol 8 Nos 2 3 4 5 1970 Vol 9 Nos 5 6 1971
- 4 *Report to the programming language committee*
CODASYL Data Base Task Group April 1971
Available through ACM
- 5 *A survey of generalized data base management systems*
CODASYL Systems Committee May 1969
Available through ACM
- 6 *Feature analysis of generalized data base management systems*
CODASYL Systems Committee May 1971
Available through ACM
- 7 M E D'IMPERIO
Information structures: Tools in problem solving
Journal of ACM SIGFIDET Vol 1 No 2 December 1969
- 8 R W ENGLER
A tutorial on data base organization
IBM Ref TR00.2004 March 1970
- 9 *Data base management system requirements*
Guide/Share
Guide Secretary Distribution (GSD-23) January 1971
A J Burris Secretary The Northern Trust Co Box N
Chicago Illinois 60690
Share Secretary Distribution (SSD-208) December 31
1970 Share Secretary
Share Inc Suite 750 25 Broadway New York New York
10004

APPENDIX

GLOSSARY

Access Method. A routine external to the application program that performs storage and retrieval of physical data.

Access Strategy. An algorithm by which stored records are identified and located.

AP. Application Program(mer)

Association. A non-directed relationship that defines a collection of zero or more units of data based on some specified criteria. The occurrences of units of data within the collection may or may not be ordered.

Binding. The firm association of an attribute of data with the application program.

Checkpoint. The act of capturing the state of units of data and those application programs operating on them for the purpose of reconstructing the data and restarting the programs.

Command. A generic term referring to either a primitive command or a compound command. The DBCL facility through which the DBM functions are invoked by an application program.

Communication Area. That area embodied within an application program that permits communication between the application program and the DBM.

Compound Command. A DBCL command which is a combination of primitive commands and procedural logic.

Data Area. That area in the application program which contains the logical data.

Data Base. A named collection of units of physical data which are related to each other in a specified manner.

Data Base Administrator (DBA). A person or persons given the responsibility for the definition, organization, protection and efficiency of the data bases for an enterprise.

Data Base Command Language (DBCL). A language whose statements are used to invoke the DBM facilities.

Data Base Descriptive Language (DBDL). A language whose statements are used to describe all units of data to the DBMS.

Data Base Directory (DBD). A collection of descriptors of all units of data that are available to the DBMS. These descriptors are derived from DBDL statements.

Data Base Management System (DBMS). The data processing system consisting of the tri-partite interaction between the Requestor, the DBA, and the DBM.

Data Base Manager (DBM). A combination of hardware and software which controls and processes all requests for data in the data bases.

Data Element. The smallest named unit of physical data stored on some medium.

Data Independence. The concept of separating the definitions of logical and physical data such that application programs need not be dependent on where or how physical units of data are stored.

Data Integrity. The concept that all units of data must be protected against accidental or deliberate invalidation.

Data Item. A unit of logical data which can be either an elementary data item or a group data item.

Data Name. A name given to units of data for the purpose of uniquely identifying that unit of data.

Data Set. A named collection of stored records of one or more stored record types. More precisely, a collection of one or more space and form extents.

DB. Data Base(s).

DBA. Data Base Administrator.

DBCL. Data Base Command Language.

DBD. Data Base Directory.

DBDL. Data Base Descriptive Language.

DBM. Data Base Manager.

DBMS. Data Base Management System.

Derived Data Element. A data element whose value is derived from the values of other data elements by a specified algorithm.

Descriptors. The detailed definition of all units of data as represented in the DBD.

Elementary Data Item. The smallest named unit of logical data available to a program.

Elementary Entity Field. A named field which refers to an attribute of an entity.

Entity. A person, place, thing or event of interest to the enterprise.

Entity Construct. An association of entity record types and the entity relationships which connect them. An entity construct when named becomes another entity record type.

Entity Field. A generic term referring to either an elementary entity field or a group entity field.

Entity Record Type. A named collection of entity field names that represents the attributes unique to a particular entity type.

Entity Relationships. A named relationship that exists between two entities of the same or different entity types.

Entity Type. A particular kind of entity. For example: employees, departments, vendors, sales, etc.

Exclusive Control. A facility to prevent multiple concurrent interactions with a specific unit of data in the data base such that the integrity of the data is preserved.

File. named collection of occurrences of logical records which may be of more than one logical record type.

Form Extent. That portion of a data set wherein all stored record occurrences are of the same stored record type.

Format. A formal description of a unit of data that

contains information about its length, base, scale, precision, representation, etc.

Group Data Item. A named association of data items.

Group Entity Field. A named association of entity fields within an entity record type.

Hierarchy. A set of directed relationships between two or more units of data; such that some units of data are considered owners while others are members. This is distinguished from a network in that in a hierarchy, each member can have one and only one owner.

Host Language. The programming language used to write the application program. A program written using this language contains DBCL statements that invoke DBM functions.

Identifier. A unit of data whose value uniquely identifies an occurrence of that unit of data or a different unit of data.

Journal. A record of all environmental conditions and changes relative to the data bank. It may include time and date stamps, user identification, attempted security breaches, changes to a data base, etc.

Linkage. A mechanism for connecting one unit of data to another.

Logical Data. That data which the application program presents to or receives from the DBM.

Logical Record. A collection of one or more data item values. More specifically, an occurrence of a logical record type.

Logical Record Type. A specific collection of one or more data items.

Logical Relationship. The relationship that exists between two units of logical data.

Materialize. The act of making logical data available to the application program.

Member. See membership

Membership. A directed relationship connecting one unit of data called the owner to another unit of data called the member.

Network. A set of directed relationships between two or more units of data such that some units of data are considered owners while others are members. Unlike a hierarchy, each member may have more than one owner.

Occurrence. A specific representation of the value of a unit of data that is usually associated with a value called its identifier.

Own Code. Specialized routines developed by an installation to perform functions not provided for by the Data Base Manager.

Physical Data. That data which the Data Base Manager stores on, or retrieves from, some medium.

Physical Record. That data accessed by the hardware from some medium with a single access.

Physical Relationship. The relationship that exists between two or more units of physical data.

Primitive Command. A DBCL command which can be executed without further intervention by the application program.

Profile. A subset of descriptors in the DBD that relates to something of interest to the DBA such as a program or a user.

Qualifier. Criteria used to select a logical record.

Real Data. A logical unit of data that has a physical unit of data as a counterpart.

Redundancy. A situation where there are multiple occurrences of a particular unit of data in a data base.

Relationships. A generic term referring in this document to either one of two kinds of relationships: membership and association.

Requestor. An individual desiring to use the data in the DBMS.

Security. The concept that units of data can be altered, viewed or processed only by users who have the proper authority and the "need to know".

Self Defining Data. A unit of data whose description appears with its occurrence.

Shared Control. A facility that allows multiple concurrent interactions with a specific unit of data in a data base.

Space Extent. A unit of allocated space on some medium which is contiguous.

Span of Control. That period of time during which the application program has exclusive or shared control of a specified unit of data.

Stored Record. A collection of one or more data element values. More specifically an occurrence of a stored record type.

Stored Record Type. A specific collection of one or more data elements.

Structure. A generic term which refers to the aggregation of units of data, their formats and their relationships.

Unit of Data. A generic term denoting a named conceptualization of logical data and physical data. The units of logical data are:

- data item
- logical record
- logical relationship
- file

The units of physical data are:

- data element
- stored record

- physical record
- physical relationship
- data base
- form extent
- space extent
- data set

User. A generic term referring to the Requestor or the DBA.

User Exit. A DBM facility that permits an installation to execute its own code.

Virtual Data. A logical unit of data that is materialized and does not exist as a physical unit of data.

User engineering principles for interactive systems*

by WILFRED J. HANSEN

Argonne National Laboratory
Argonne, Illinois

INTRODUCTION

The 'feel' of an interactive system can be compared to the impressions generated by a piece of music. Both can only be experienced over a period of time. With either, the user must abstract the structure of the system from a sequence of details. Each may have a quality of 'naturalness' because successive actions follow a logically self-consistent pattern. Finally, a good composer can write a new pattern which will seem, after a few listenings, to be so natural the observer wonders why it was never done before.

Just as a composer follows a set of harmonic principles when he writes music, the system designer must follow some set of principles when he designs the sequence of give and take between man and machine. This paper reports a set of principles—called user engineering principles—which were employed while designing the Emily text editing system. These principles evolved during the course of the project, but were originally based on the author's experiences with a number of other text editing systems.^{2,3,4,5}

In text editing applications, the user sits at a console and creates, views, or modifies a document, be it program, speech, article or a chapter of his next book. Here the computer is a tool for the creative worker and the emphasis must be on capturing his thoughts with minimal interference. More common in commercial environments are interactive systems designed as tools to coordinate the work of many clerical workers. Examples are order entry, point-of-sale, inventory control, defense surveillance, and the like. The principles outlined below, though originally intended for creative work, are equally applicable to clerical work. Sometimes more so, because clerks may not have the commitment of the creative worker.

One restriction on a few of the principles below is that they apply to systems with display devices for output. This is essential, because a basic principle is that the system respond to the user as fast as possible. A visual display can present more information in less time than available hardcopy devices. The 'economy' of the terminal device must be weighed against the cost of attention-wander-time as the user interacts with the system. Other than the terminal, cost is not a problem in the application of these user engineering principles. In general, they dictate features that are inexpensive to design into a system. They are, however, often expensive to include after implementation is under way.

Disciplines similar to user engineering have been called human engineering, human factors, and ergonomics, but these terms most often refer to analog systems like airplane cockpits where the pilot guides a process. User engineering applies to digital systems where the goal is to store or retrieve information. D. Engelbart⁶ refers to these principles as 'User Feature Design.' His point is that this term emphasizes that the features are being designed for the user rather than the other way around. In fact, though, any interactive system will require retraining of the users and some systems—like Emily—may require the user to alter thinking habits of many years standing. (But let there be no mistake, the author is deeply committed to a policy of modifying the system to fit the user.) Other sets of user engineering principles have been reported by L. B. Smith⁷ and J. G. Mitchell.⁸ Their suggestions are compatible with those below, but less comprehensive. The reader should also read R. B. Miller's paper⁹ in which he attempts to estimate a maximum permissible response time in seventeen interactive contexts.

The user engineering principles in the second section below are illustrated by reference to the Emily text editing system. For this reason, the Emily system is sketched in the first section. More complete descrip-

* The work reported here was supported by the U.S. Atomic Energy Commission. The text is taken from the second and fourth chapters of the author's thesis.¹

```

1  <STMT> : DO <ARITHV> = <ARITHX> TO
      <ARITHX>; <STMT*> END;
2      : <ASGN STMT>
3  <STMT*> : <STMT>
4  <ASGN STMT> : <ARITH> = <ARITHX>;
5  <ARITHX> : <ARITH>
6      : <ARITHV>
7      : <NUMBER>
8      : <ARITHX> + <ARITHX>
9  <ARITHX*> : <ARITHX>
10 <ARITHV> : <ARITH>
11      : <ARITH> (<ARITHX*>)
12 <ARITH> IS AN IDENTIFIER
13 <NUMBER> IS A CONSTANT
    
```

Figure 1—Portion of syntax for PL/I

Each rule specifies a possible replacement for the non-terminal to the left of the colon. If the left side is omitted, it is the same as the previous line. Rules 12 and 13 specify special classes of terminal symbols

tions are available elsewhere.^{10,11} Emily has been implemented for an IBM 2250 Graphic Display Unit, model 3. The 2250 displays lines and characters on a 12" by 12" screen. The user can give commands to the system with a light pen, a program function keyboard, and an alphameric keyboard.

THE EMILY SYSTEM

Emily is primarily intended for construction and modification of computer programs written in higher level languages. Many such systems exist, but all existing systems require the programmer to enter his text as a sequence of characters. With Emily, the user constructs his text by selecting choices from the menu to replace certain symbols in the text. For example, the symbol <STMT> might be replaced by

```

DO <ARITHV> = <ARITHX> TO <ARITHX>;
  <STMT*>
END;
    
```

Replaceable symbols begin with '<', end with '>', and contain a name that usually has some relation to the meaning of the string generated by the symbol. Such symbols are called *non-terminal symbols*, because of their role in the Backus-Naur Form (BNF) notation for describing programming languages.¹²

In BNF a syntax for a formal language has three parts—a set of *terminal symbols*, a set of *non-terminal symbols*, and a set of *syntactic rules*. The terminal symbols are those characters and strings of characters (punctuation, reserved words, identifiers, constants) that can be part of the completed text. The non-terminal symbols are a specific set of symbols introduced only to help describe the structure of the formal language. Every non-terminal symbol must be replaced by terminal symbols before the entire text is complete,

```

          <STMT>                                1
DO <ARITHV> = <ARITHX> TO <ARITHX>;          10
  <STMT*>
END;
DO <ARITH> = <ARITHX> TO <ARITHX>;          12
  <STMT*>
END;
DO I = <ARITHX> TO <ARITHX>;                7
  <STMT*>
END;
. . .                                       13,7,13,3,2
DO I = 1 TO 20;                             4
  <ASGN STMT>
END;
. . .                                       12,8,5,12,6,11,
                                          12,9,5,12
DO I = 1 TO 20;
  S = S + A(I);
END;
    
```

Figure 2—Steps in the generation of a DO loop
In each step, the non-terminal in the rectangle is replaced according to the rule whose number appears at the right

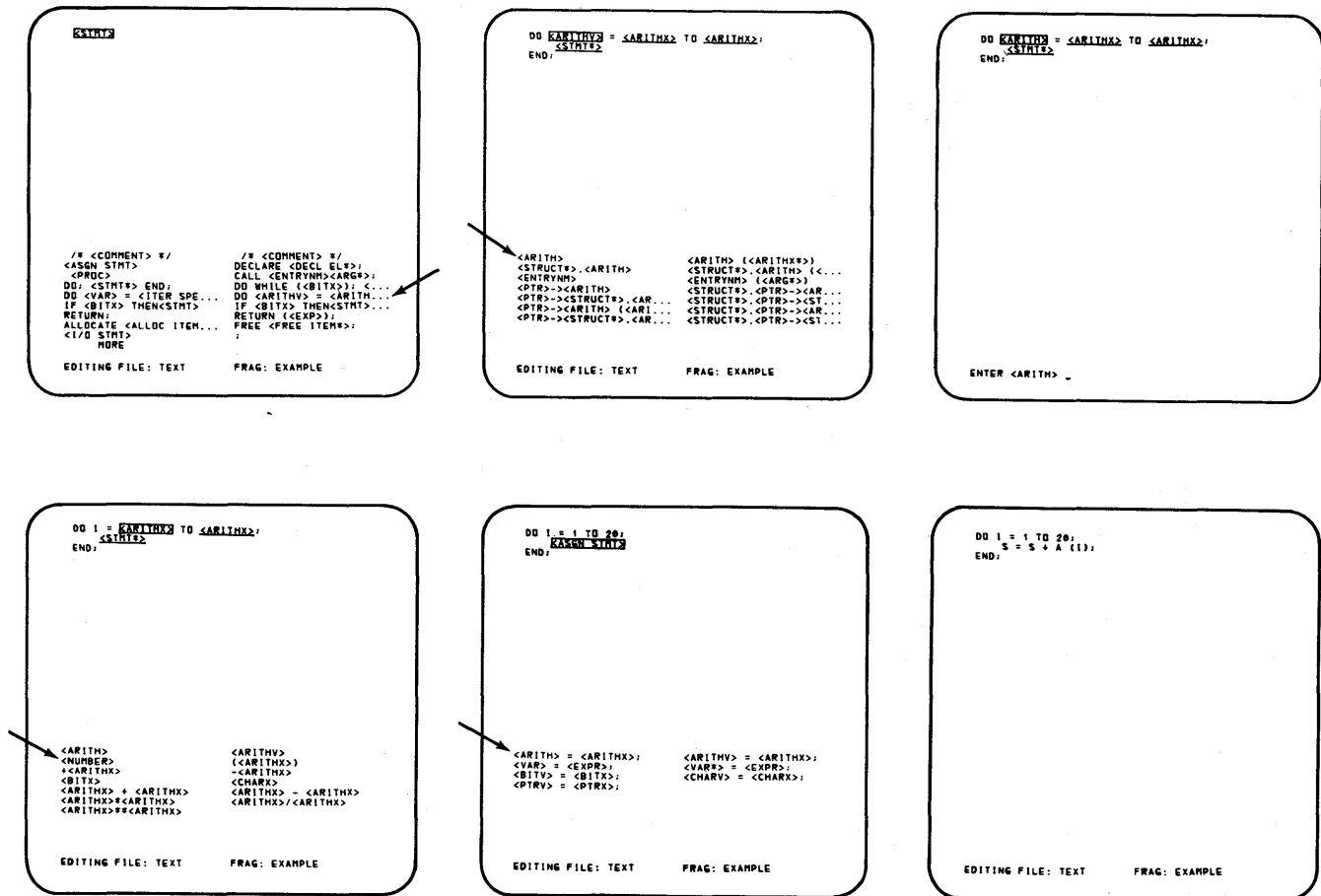


Figure 3—Generation of a DO loop with Emily

These photographs show the same steps as shown in Figure 2. The menu displays all the choices available in the implemented PL/I syntax. An arrow indicates the syntax rule the user will select next. Up to twenty-two lines of text may be shown in the text area, so it appears empty with only 3 lines

but the only allowable replacements for a given non-terminal are specified by the syntactic rules. In each rule, the given non-terminal is on the left followed by a colon followed by the sequence of symbols that may replace the non-terminal. As an example, Figure 1 shows a portion of the syntax for PL/I. Figure 2 shows a DO loop generated using this syntax.

It is important to note that a string generated according to a syntax is not simply a sequence of characters, but can be divided into hierarchies of substrings on the basis of the syntactic rules. Each non-terminal in the sequence of symbols for a rule generates a subsequence. The DO statement in Figure 2 can be one of a sequence of statements in some higher DO loop and can also contain a subordinate sequence of statements (generated by <STMT*>). Replacement of a non-terminal by a rule can be thought of as replacing the non-terminal with a pointer to a copy of the rule.

The non-terminals in this copy can be further replaced by pointers to copies of other rules. In a diagram each syntactic rule used in the generation of the string is represented by a *node* (a rectangle). The node contains one pointer to a subordinate node for each non-terminal in the syntactic rule. The subordinate node is called a subnode or a descendant, while the pointing node is called the parent.

Emily text structure

Text in the Emily system is stored in a *file*, which may contain any number of *fragments*. Each fragment has a name and contains a piece of text generated by some non-terminal symbol. Generated text is physically stored in a hierarchical structure like that described above. Each node is a section of memory containing (a) the number of the syntax rule for which this node

was generated, and (b) one pointer to each subnode. In a completed text, there is one descendant node for each non-terminal in the syntax rule and the pointer to a descendant is the address of the section of memory where it is stored. If no text has been generated for a non-terminal symbol, there is no subnode and the corresponding pointer is replaced by a code representing the non-terminal symbol. If a subnode of a node is an identifier, the pointer points at a copy of the identifier in a special area. All pointers at a given identifier point to the same copy in this identifier area. Other than identifiers, each node is pointed at exactly once within the text structure. This guarantees that if a node is modified, only one piece of text is affected.

Notice that punctuation and reserved words do not appear in this representation of text. Instead, they can be generated because the syntax rule number identifies the appropriate rule. Two tables in Emily contain coded forms of the syntax rules. One table, called the *abstract syntax*, controls the hierarchical structure of generated text. It specifies which syntax rules can replace a given non-terminal symbol and the sequence of non-terminal symbols on the right-hand-side of each syntax rule. Another table, the *concrete syntax*, tells how to display each rule; it includes punctuation, reserved words, and formatting information like indentation and line termination.

Creating text

The Emily user creates hierarchical text in a series of steps very similar to Figure 2. In each step the right side of a rule is substituted for a non-terminal symbol. Before the user creates any text, the fragment contains a single non-terminal symbol. In the case of Figure 2, that symbol is $\langle \text{STMT} \rangle$. The user sees the result of each step on the 2250 display. Figure 3 shows the steps of Figure 2 as they appear on the screen.

While using the Emily system the 2250 screen appears to be divided into three areas: text, menu, and message. The text area occupies the upper two-thirds of the screen and displays the text the user is creating. The lower third of the screen is the menu where Emily displays the strings the user can substitute in the text. The bottom line of the screen is the message area, where Emily requests operands and displays status and error messages.

Non-terminal symbols** in the text area are underlined to make them stand out. One of the non-terminals

** When it is displayed, a non-terminal is the end (or terminal) of a branch of the hierarchical structure. It is called a non-terminal because it must be replaced with a string of terminals before the text is complete.

is the *current non-terminal* and is surrounded by a rectangle. The menu normally displays all strings that can be substituted for the current non-terminal. These strings are simply the right sides of the syntax rules that have the current non-terminal on the left.

When the user points the light pen at an item in the menu Emily substitutes that item for the current non-terminal. Usually, the substitution string contains more than one non-terminal and the new current non-terminal is the first of these. The user can also change the current non-terminal by pointing the light pen at any non-terminal in the display. Emily moves the rectangle to that non-terminal and changes the menu accordingly. When the current non-terminal is an identifier, the menu displays identifiers previously entered in the required class (some of the classes for PL/I are $\langle \text{ARITH} \rangle$, $\langle \text{CHAR} \rangle$, and $\langle \text{ENTRYNM} \rangle$). The user may select one of these, or he may enter a new identifier from the keyboard. Constants are also entered from the keyboard.

Viewing text

Since text is stored hierarchically within Emily, it can be viewed with operations that take advantage of that structure. The user may wish to descend into the structure and examine the details of some minor substructure. Alternatively, he may wish to view the highest levels of the hierarchy with substructures represented by some appropriate symbol. Both of these viewing operations are possible with Emily.

The symbol displayed to represent a substructure is called a *holophrast*. This symbol begins and ends with an exclamation mark and contains two parts separated by a colon. The first part is the non-terminal symbol that generated the substructure and the second part is the first few characters of the represented string. Figure 4 shows three examples of holophrasts. Note that contraction to a holophrast only changes the view of the file and it does not modify the file itself. Moreover, the user never enters a holophrast from the keyboard; they are displayed only as a result of contracting text.

The user contracts a structural unit in the display by pushing a button on the program function keyboard and then pointing at some character in the text. The selected character is part of the text generated by some node in the hierarchical structure. The display of this node is replaced by a holophrast. If the user points at a holophrast, the father of the indicated node contracts to a holophrast which subsumes the earlier one. To expand a holophrast back to a string, the user returns to normal text construction mode and points the light pen at the holophrast.

The operations to ascend and descend in the text hierarchy are also invoked by program function buttons. To descend in the hierarchy the user pushes the IN button and points at a part of the text. The selected node becomes the new *display generating node*; subsequent displays show only this node and its subnodes. The OUT button lets the user choose among the ancestors of the display generating node and then makes the selected ancestor the new display generator.

System environment

At Argonne National Laboratory, the 2250 is attached to an IBM 360 model 75. The 75 is under control of the MVT version of OS/360. Unit record input/output is controlled by ASP in an attached 360/50. The 360/75 has one million bytes of main core and one million bytes of a Large Capacity Storage Unit.

The Emily system itself requires 60K bytes of main core (the maximum permitted for a 2250 job at Argonne) and about 400K bytes of LCS. Emily is written in PL/I and uses the Graphic Subroutine Package to communicate with the 2250. Files for Emily are stored on a 2314 disk pack. Emily is table driven and can manipulate text in any formal language. To date, tables have been created for four languages: PL/I,

GEDANKEN,¹³ a simple hierarchy language for writing thesis outlines, and a language for creating syntax definitions.

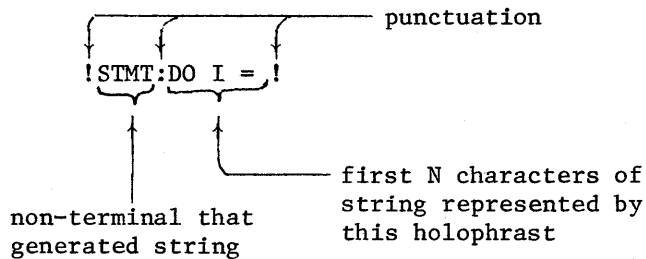
USER ENGINEERING PRINCIPLES

The first principle is **KNOW THE USER**. The system designer should try to build a profile of the intended user: his education, experience, interests, how much time he has, his manual dexterity, the special requirements of his problem, his reaction to the behavior of the system, his patience. One function of such a profile is to help make specific design decisions, but the designer must be wary of assuming too much. Improper automatic actions can be an annoying system feature.

A more important function of the first principle is to remind the designer that the user is a human. He is someone to whom the designer should be considerate and for whom the designer should expend effort to provide conveniences. Furthermore, the designer must remember that human users share two common traits: they forget and they make mistakes. With any interactive system problems will arise—whether the user is a high school girl entering orders or a company president asking for a sales breakdown. The user will forget how to do what he wants, what his files contain, and even—if interrupted—what he wanted to do. Good system design must consider such foibles and try to limit their consequences. The Emily design tried to limit these consequences by explicitly including a fallible memory and a capacity for errors in the intended user profile. Other characteristics assumed are:

- curious to learn to use a new tool,
- skilled at breaking a problem into sub-problems,
- familiar with the concept of syntax and the general features of the syntax for the language he is using,
- manually dextrous enough to use the light pen,
- not necessarily good at typing.

Throughout the following discussion, reference is made to 'modularity' and 'modular design.' These terms refer to the structure of the program, but have important consequences for user engineering. A modular program is partitioned into subroutines with distinct functions and distinct levels of function. For instance, a high level modular subroutine implements a specific user command but modifies the data structure only by calls on lower level modules. To be useful for the general case, the lower modules must have no functions dependent on specific user commands. In the Emily system, for example, there are user commands to MOVE and COPY text and there are low level routines



```
DO I = 1 TO 20;
  !STMT:S = S + !
END;
```

```
DO I = 1 TO 20;
  S = !ARITHX:S + A(I);
END;
```

Figure 4—Examples of holoprasts

All three examples show the DO loop, but each has been contracted differently. The user may change N, the number of characters of the substring. In the examples, N is seven

User Engineering Principles

First principle: Know the user

Minimize Memorization

- Selection not entry
- Names not numbers
- Predictable behavior
- Access to system information

Optimize Operations

- Rapid execution of common operations
- Display inertia
- Muscle memory
- Reorganize command parameters

Engineer for Errors

- Good error messages
- Engineer out the common errors
- Reversible actions
- Redundancy
- Data structure integrity

Figure 5—User engineering principles

for the same functions. These low level routines always destroy the existing information at the destination, but the user commands are defined to move that existing information to the special fragment *DUMP*. The low level routines must be called twice (destination→*DUMP* ; source→destination) to implement the user commands, but these same routines are used in several other places in the system. Designing adequate modularity into a system requires careful planning at an early stage, but pays off with a system that takes less time to implement, is easier to modify, and can be debugged with fewer problems and more confidence of success.

Specific user engineering principles to help meet the first principle can be categorized into

MINIMIZE MEMORIZATION,
OPTIMIZE OPERATIONS,
ENGINEER FOR ERRORS.

The principles are outlined in Figure 5.

Minimize memorization

Because the user forgets, the computer memory must augment his memory. One important way this can be accomplished is by observing the principle SELECTION NOT ENTRY. Rather than type a character string or operation name, the user should select the appropriate item from a list displayed by the computer. In a sense, the entire Emily system is based on this principle. The user selects syntax rules from the menu and never types text. Even when an identifier is to be entered, Emily displays previously entered identifiers; though the user must type in new identifiers. Because the system is presenting choices, the user need not remember the exact syntax of statements in the language, nor the spelling of identifiers he has declared. Moreover, each selection—a single action by the user—adds many characters to the text. Thus if the system can keep up with the user, he can build his text more quickly than by keyboard entry.

The principle of 'selection not entry' is central to computer graphics and by itself constitutes a revolution in work methods. The author first saw the principle in the work of George¹⁴ and Smith⁷ but has since observed it in many systems. The fact is that a graphic display—attached to a high bandwidth channel—can display many characters in the time it would take a user to type very few. If the choices displayed cover the user's needs, he can enter information more quickly by selection. Ridsdale¹⁵ has reported a patient note system used in a British hospital that is based on the principle of selection. In this system, selection is not by light pen but by typing the code that appears next to the desired choice in the menu.

Experience with Emily suggests that keyboard code entry is better than light pen selection because of two user frustrations. First, the menu does not provide a target for the light pen while the display is changing; and second, the delay can vary depending on system load. With keyboard codes, the user can go at full speed in making selections he is familiar with, but when he gets to unfamiliar situations he can slow down and wait for the display. Thus, his behavior can travel the spectrum from typing speed to machine paced selection.

The second principle to avoid memorization is NAMES NOT NUMBERS. When the user is to select from a set of items he should be able to select among them by name. In too many systems, choices are made by entering a number or code which the system uses to index into a set of values. Users can and do memorize the codes for their frequent choices, though this is one more piece of information to obscure the problem at

hand. But when an uncommon choice is needed, a code book must be referenced. Symbol tables are understood well enough that there is no excuse for not designing them into systems so as to replace code numbers with names. In Emily, there are names for files, fragments, display statuses, syntaxes, and non-terminals. Conceivably, the user could even supply a name to be displayed in each holophrast. In practice, though, so many holophrasts are displayed that the user would never be done making up names. For this reason, the holophrast contains the non-terminal and the first few characters of the text—a system generated 'name' with a close relation to the information represented by that name.

It is also possible to forget the meaning of a name, so a system should also provide a dictionary. System names should be predefined and the user should be allowed to annotate any names he creates. The lack of a dictionary in Emily has sometimes been a nuisance while trying to remember what different text fragments contain.

The next principle, PREDICTABLE BEHAVIOR, is not easy to describe. The importance of such behavior is that the user can gain an 'impression' of the system and understand its behavior in terms of that impression. Thus by remembering a few characteristics and a few exceptions, the user can work out for himself the details of any individual operation. In other words, the system ought to have a 'Gestalt' or 'personality' around which the user can organize his perception of the system. In Emily all operations on text appear to make it expand and contract. Text creation expands a non-terminal to a string and the viewing operations expand and contract between strings and holophrasts. This commonality lends the unity of predictable behavior to Emily.

Predictable behavior is also enhanced by system modularity. If the same subroutine is always used for some common interaction, the user can become accustomed to the idiosyncracies of that interaction. For instance, in Emily there is one subroutine for entering names and other text strings so that all keyboard interactions follow the same conventions.

The last memory minimization principle is ACCESS TO SYSTEM INFORMATION. Any system is controlled by various parameters and keeps various statistics. The user should be given access to these and should be able to modify from the console any parameter that he can modify in any other way. With access to the system information, the user need not remember what he said and is not kept in the dark about what is going on. Emily provides means of setting several parameters, but fails to have any mechanism for displaying their values. This oversight is due to a failure to remember

that the user might not have written the system. Another such oversight is a failure to provide error messages for many trivial user errors. Even worse, the 'MULTIPLE DECLARATION' error message originally failed to say which identifier was so declared. This has been corrected, but should have been avoided by attention to the 'Access to system information' principle of user engineering.

Optimize operations

The previous section stressed the design—the logical facilities—of the set of commands available to the user. 'Optimize operations' stresses the physical appearance of the system—the modes and speeds of interaction and the sequence of user actions needed to invoke specific facilities. The guiding principle is that the system should be as unobtrusive as possible, a tool that is wielded almost without conscious effort. The user should be encouraged to think not in terms of the light pen and keyboard, but in terms of how he wants to change the displayed information.

The first step in operation optimization is to design for RAPID EXECUTION OF COMMON OPERATIONS. Because Emily text is frequently modified in terms of its syntactic organization, a data structure to represent text was chosen so as to optimize such modification. The text display is regenerated frequently, so considerable effort was expended to optimize that routine. More effort is required, though; it is still slow largely because a subroutine is called to output each symbol. Less frequent operations like file switching do not justify special optimization. Lengthy operations, however, should display occasional messages to indicate that no difficulty has occurred. For instance, while printing a file Emily displays the line number of each tenth line as it is printed.

As the system reacts to a user's request, it should observe the principle of DISPLAY INERTIA. This means the display should change as little as necessary to carry out the request. The Emily DELETE operation replaces a holophrast (and the text it represents) with a non-terminal symbol. The size and layout of the display do not change drastically. Text cannot be deleted without first being contracted to a holophrast, thus deletion—a drastic and possibly confusing operation—does not add the disorientation of a radically changed display. The Emily display also retains inertia in that the top line changes only on explicit command. Some linear text systems always change the display so the line being operated on is in the middle of the display. Because the perspective is constantly shifting, the user

is sometimes not sure where he is. The Emily automatic indentation provides additional assistance to the user. As text is created in the middle of the display, the bottom line moves down the display. Since this line is often not indented as far as the preceding line, its movement makes a readily perceptible change in the display.

One means of reducing the user's interaction effort is to design the system so the user can operate it on 'MUSCLE MEMORY.' Very repetitive operations like driving a car or typing are delegated by the conscious mind to the lower part of the brain (the medulla oblongata). This part of the brain controls the body muscles and can be trained to perform operations without continual control from the conscious mind. One implication of muscle memory is that the meaning of specific interactions should have a simple relation to the state of the system. A button should not have more than a few state dependent meanings and one button should be reserved to always return the system to some basic control state. With such a button, the muscle memory can be trained to escape from any strange or unwanted state so as to transfer to a desired state. In Emily the buttons of the program function keyboard obey these principles. The NORMAL button always returns the entire system to a basic state waiting for commands. Other buttons have very limited meanings and it is almost always possible to abort one command and invoke another simply by pushing the other button (without pushing NORMAL first).

A second implication of muscle memory for system design is that the system must be prepared to accept commands in bursts exceeding ten per second. (Typing 100 words per minute is 10 characters per second. A typing burst can be faster.) It is not essential that the system react to commands at this rate, because interactive computer use is characterized by command bursts followed by pauses for new inspiration. But if command bursts are not accepted at a high rate, the muscle memory portion of the brain cannot be given full responsibility for operations. The conscious brain has to scan the system indicators waiting for GO. Command bursts from muscle memory account for the unsuitability of the light pen for rule selection as discussed under 'selection not entry.'

In addition to optimizing the interaction time, the system designer must be prepared to REORGANIZE COMMAND PARAMETERS. Observation of users in action will show that some commands are not as convenient as their frequency warrants while other commands are seldom used. Inconvenient commands can be simplified while infrequent commands can be relegated to sub-commands. Such reorganization is simplified if the origi-

nal system design has been adequately modularized. High level command routines can be rewritten without rewriting low level routines and the latter can be used without fear that they depend on the higher level.

A good example of command reorganization in Emily has been the evolution of the view expansion commands. In the earliest version, pointing the light pen at a holophrast expanded it one level, so that each of the sub-nodes of the holophrast became a new holophrast. With this mechanism, many interactions were required to view the entire structure represented by a holophrast. Very soon the system-designer/user added a system parameter called 'expansion depth.' This parameter dictated how many levels of a holophrast were to be expanded. To set the expansion depth, the user pushed a button (on the program function keyboard) and typed in a number (on the alphameric keyboard). It soon became obvious that users almost always set the expansion depth to either one or all. Consequently, two buttons were defined, so that the user could choose either option quickly. Later, the button for typing in the expansion depth was removed and that function placed under a general 'set parameters' command. Further experience may show that only the 'expand one level' button is required. It would take effect only during the next holophrast expansion. At all other times, holophrasts would always be expanded as far as possible.

Engineer for errors

Modern computers can perform billions of operations without errors. Knowing this, system designers tend to forget that neither users nor system implementers achieve perfection. The system design must protect the user from both the system and himself. After he has learned to use a system, a serious user seldom commits a deliberate error. Usually he is forgetful, or pushes the wrong button without looking, or tries to do something entirely reasonable that never occurred to the system designer. The learner, on the other hand, has a powerful, and reasonable, curiosity to find out what happens when he does something wrong. A system must protect itself from all such errors and, as far as possible, protect the user from any serious consequences. The system should be engineered to make catastrophic errors difficult and to permit recovery from as many errors as possible.

The first principle in error engineering is to provide GOOD ERROR MESSAGES. These serve as an invaluable training aid to the learner and as a gentle reminder to the expert. With a graphic display it is possible to pre-

sent error messages rapidly without wasting the user's time. Error messages should be specific, indicating the type of error and the exact location of the error in the text. Emily does not have good messages for user errors. Currently, the system blows the whistle on the 2250 and waits for the next command from the user. Each error is internally identified by a unique number, and it will not be difficult to display the appropriate message for each number.

It is not enough to simply tell the user of his errors. The system designer must also be told so he can apply the principle **ENGINEER OUT THE COMMON ERRORS**. If an error occurs frequently, it is not the fault of the user, it is a problem in the system design. Perhaps the keyboard layout is poor or commands require too much information. Perhaps consideration must be given to the organization of basic operations into higher level commands.

Emily provides several means of feedback from the user to the system designer. (Though for the most part, they have been one and the same.) A log is kept of all user interactions, user errors, and system errors. There is a command to let the user type a message to be put in the log and this message is followed by a row of asterisks. When the user is frustrated he can push a 'sympathy' button. In response, Emily displays at random one of ten sympathetic messages. More importantly, frustration is noted in the log and the system designer can examine the user's preceding actions to find out where his understanding differed from the system implementation.

'Engineering errors out' does not mean to make them impossible. Rather they should be made sufficiently more difficult that the user must pause and think before he errs. In Emily, time consuming operations like file manipulation always ask the user for additional operands. If he does not want the time consuming operation he can do something else. To delete text, the user must think and contract it to a holophrast. This means that large structures cannot be cavalierly deleted.

A single erroneous deletion can inadvertently remove a very large substructure from the file. To protect the user the system must provide **REVERSIBLE ACTIONS**. There ought to be one or more well understood means for undoing the effects of any system operation. In Emily, a deleted structure is moved to ***DUMP***. If the user has made a mistake, he can reach into this 'trash can' and retrieve the last structure he has deleted. (Deletion does destroy the old contents of ***DUMP***.) A more general reversible action mechanism would be a single button that always restored the state existing before the last user interaction. Emily has no

such button, but the QED system¹⁶ supplies a file containing all commands issued during the console session. The user can modify this file of commands and then use it as a source of commands to modify the original text file again.

Besides helping the user escape his own mistakes, error engineering must protect the user from bugs in the system and its supporting software. Modular design is important to such protection because it minimizes the dependencies among system routines. The implementer should be able to modify and improve a routine with confidence that his changes will affect only the operation of that routine. Even if the changes introduce bugs, the user will be protected if the designer has observed the principles of redundancy and data structure integrity.

REDUNDANCY simply means that the system provides more than one means to any given end. A powerful operation can be backed up by combinations of simpler operations. Then if the powerful operator fails, the user can still continue with his work. Such redundancy is most helpful while debugging a system, but very few systems are completely debugged and any aids to the debugger can help the user. As an adjunct of redundancy, the system must detect errors and let the user act on them, rather than simply dumping memory and terminating the run. In Emily, the PL/I ON-condition mechanism very satisfactorily catches errors. They are passed to a subroutine in Emily that tells the user that a catastrophe has occurred and names the offending module. Control then returns to the normal state of waiting for a command from the user, who has the option to continue or call for a dump.

A system should provide sufficient **DATA STRUCTURE INTEGRITY** that regardless of system or hardware trouble some version of the user information will always be available. This principle is especially applicable to Emily where most of the information is encoded by pointers. A small error in one pointer can lose a large chunk of the file. Some effort has been spent ensuring that errors in Emily will not damage the part of the data structure kept in core during execution. But if an error abruptly terminates Emily execution (such errors are generally in the system outside Emily) the file on the disk may be in a confused state. Currently, the only protection is to copy the file before changing it, but there are file safety systems that do not rely on the user to protect himself, and one of these should be implemented for Emily.

Protection and assistance for the user are keywords in user engineering. The principles outlined in this paper are not as important as the general approach of tailoring the system to the user. Only by such an ap-

proach can Computer Science divest the computer of its image as a cold, intractable, and demanding machine. Only by such an approach can the computer be made sufficiently useful and attractive to take its place as a valuable tool for the creative worker.

ACKNOWLEDGMENTS

I am grateful to Dr. John C. Reynolds and Dr. William F. Miller. Any success of the Emily project is due to their persistent advice and encouragement.

REFERENCES

- 1 W J HANSEN
Creation of hierarchic text with a computer display
Argonne National Laboratory ANL-7818 Argonne Illinois 1971
- 2 J McCARTHY D BRIAN G FELDMAN
J ALLEN
THOR—a display based time sharing system
AFIPS Conf Proc Vol 30 (SJCC) 1967 pp 623-633
- 3 W WEIHER
Preliminary description of EDIT2
Stanford Artificial Intelligence Laboratory Operating Note 5 Stanford California 1967
- 4 DEC LIBRARY
PDP-6 time sharing TECO
Stanford Artificial Intelligence Laboratory Operating Note 34 Stanford California 1967
- 5 STANFORD UNIVERSITY COMPUTATION CENTER
Wylbur reference manual
Campus Facility Users Manual Appendix E Stanford California 1968
- 6 D C ENGELBART
Private communication
Stanford Research Institute Menlo Park California 1971
- 7 L B SMITH
The use of man-machine interaction in data-fitting problems
Stanford Linear Accelerator Center Report 96 Stanford California 1969
- 8 J G MITCHELL
The design and construction of flexible and efficient interactive programming systems
Department of Computer Science Carnegie-Mellon University Pittsburgh Pennsylvania 1970
- 9 R B MILLER
Response times in man-computer conversational transactions
AFIPS Conf Proc Vol 33 (FJCC) part 1 1968 pp 267-277
- 10 W J HANSEN
Graphic editing of structured text
in *Advanced Computer Graphics* R D PARSLOW R E GREEN editors Plenum Press London 1971 pp 681-700
- 11 W J HANSEN
Emily user's manual
Argonne National Laboratory Argonne Illinois forthcoming
- 12 J W BACKUS
The syntax and semantics of the proposed international algebraic language of the Zurich ACM-GAMM conference
Proc International Conf on Information Processing UNESCO 1959 pp 125-132
- 13 J C REYNOLDS
GEDANKEN—a simple typeless language based on the principle of completeness and the reference concept
Comm ACM Vol 13 No 5 1970 pp 308-319
- 14 J E GEORGE
Calgen—an interactive picture calculus generation system
Computer Science Department Report 114 Stanford University Stanford California 1968
- 15 B RIDSDALE
The visual display unit for data collection and retrieval
in *Computer Graphics in medical research and hospital administration* R D PARSLOW R E GREEN editors Plenum Press London 1971 pp 1-8
- 16 K THOMPSON
QED text editor
Bell Telephone Laboratories Murray Hill New Jersey 1968

Computer assisted tracing of text evolution

by W. D. ELLIOTT, W. A. POTAS and A. VAN DAM

Brown University
Providence, Rhode Island

INTRODUCTION

Many situations exist in which convenient access to the detailed evolutionary information associated with a text's development is desirable:

- (1) The principal author of a paper edited by a number of people, perhaps within the context of a group project, might desire to see who made what changes and comments.
- (2) Many periodicals retain the entire evolutionary development of each article, so that all of the changes leading to the development of a final version can be identified in case, for example, a resulting article is legally challenged.
- (3) There are many instances when an author would like to be able to determine the exact nature of his editing changes. One example is that of an author who wants to study how his thoughts on a particular issue have evolved and matured over a period of time or who wants to determine how an error has crept into his writing. Another example is the case of an individual who has written a computer program (a particular kind of text) for use on one computer system and who must then convert the program to run it on another system. If he later improved this new edition and then discovered he had to run the improved program under the old system, some manner of noting all the individual changes made to the original would be important in order to separate the conversion changes from the improvement changes.

Conventionally, manuscripts are written by editing a series of hard copy rough drafts until a final version is produced, where the creation of each new draft typically entails many changes of wide-ranging complexity. If a writer desires to compare the changes between two different drafts for phraseology or content, wishes to

review the manner in which his treatment of a topic has changed during the course of editing, or must determine how or when during editing an error has been introduced into his text, his marked-up hard copy drafts (if they have been saved) provide a rough means of tracing the way in which his text developed. Several other attributes of hard copy are the portability of hard copy drafts, the ability to easily access material anywhere within these drafts, and the ability to identify different editors or distinguish between different editing passes on the basis of handwriting style or color of ink.

The principal disadvantage of hard copy editing is its tendency to lead to congestion when many changes are made to a single draft. As a result, a writer may be inhibited from making further editing changes to a draft because of the increase in information density on a page which is fixed in size. Fixed page size also encumbers a writer in making editing changes which involve points on different pages, such as moving or copying text from one place to another. Furthermore, the changes indicated on any single draft may be sufficiently convoluted that the evolutionary nature of the changes may become difficult to discern. A writer faced with such a mass of editing changes could either continue to edit the same draft, in which case the text would become even more illegible, or he could frequently start new drafts to insure a legible presentation of the evolutionary information. Creation of a plethora of drafts would, however, impair later access to the developmental changes because of the necessity of searching through the great number of drafts.

The conventional method employed to deal specifically with evolutionary information uses revision bars, vertical lines placed in the margin of a text to call attention to newly made changes. Although revision bars show the points of change more clearly than heavily edited hard copy drafts, evolutionary information associated with overlapping editing changes is obscured. For example, whenever several editing changes within a

given portion of the text overlap, the revision bars for the edits merge and the distinction between the individual editing changes is lost. A more significant drawback to revision bars is the loss of all deleted text, any annotative information explaining editing changes, and information concerning the description of the individual edits (the context in which changes were made, in particular). This information could be partially preserved by annotating the revision bars with information specifying what the indicated changes were, who made each change, and when each was made and why. (As an example, D. C. Englebart's NLS computer assisted editing system¹ provides a facility for storing the initials of the last person to edit a given statement as well as the date and the time of the editing session.) However, the loss of any deleted material and the general inability to see exactly what changes were made without comparing separate drafts remain inherent problems in any revision bar system.

Computer based editing systems such as NLS and HES² have been developed which provide a writer with excellent facilities for online composition and editing of his text. Despite their high degree of sophistication, none of these systems have the ability to retain any significant degree of evolutionary information concerning a text's development. Except for Englebart's NLS system which provides the limited facilities noted above, the only manner in which text editing systems provide evolutionary information concerning a text's development is by allowing printing or offline storage of intermediate drafts which must be manually compared in order to identify specific changes.

Although hard copy, revision bar, and current computer aided editing systems cannot individually provide optimal capabilities for both editing and evolutionary tracing, each does have its own particular advantages. It is the purpose of this paper to consider how facilities for tracing the evolution of a changing text can be integrated with the facilities provided by advanced currently existing computer aided text editing systems to produce a more comprehensive writing system.

A COMPUTER AIDED WRITING SYSTEM WITH EVOLUTIONARY TRACING

In contrast with the above methods, the next system to be considered is capable of preserving the finely delineated details of a text's evolution without requiring the retention of separate drafts. This system, computerized of necessity and using an interactive graphics terminal for system/user interface, has been implemented by the authors. The record of a text's evolutionary development is accumulated by storing a com-

plete description of each editing operation, as the operation is performed, into the system's internal data structure representation of a text. For example, text which an author has deleted and which does not subsequently appear on the display screen is retained in the internal data structure as part of the system entry for that change. The system identifies all changes by editor and date, and allows an editor to optionally associate explanatory annotation with any change.

The online storage and CPU requirements needed to support a system providing access to the evolutionary development of a text are of necessity quite large, since each edit recorded in the system requires that a significant amount of information be stored, including all "deleted" material, and since the total amount of information grows commensurately with the number of editing operations performed. Additionally, the display is called upon to present a substantial amount of information to the user, resulting in a relatively dense presentation. Although the system has its inherent shortcomings, it does possess two distinct advantages over currently available editing methods—computer assisted editing which offers both convenient editing functions (including insert, delete, substitute, move and copy) and immediate display of the updated text, and use of the accumulated evolutionary information to provide both *passive review* facilities for examining prior editing changes and a set of more advanced *active review* facilities which enable the user to modify specific editing changes *ex post facto*.

PASSIVE REVIEW

Although this system is able to accumulate a complete record of editing changes made to a text, the measure of the system's usefulness is its ability to present this information to the user so that he can easily deal with the text and its stored evolutionary information. The system facilities described below provide an "instant replay" capability for displaying in any of the basic display modes the development of selected portions of a text, with a number of options available to filter the material to be displayed. *Normal display* always shows a clean, updated version of a text, incorporating all applicable changes, while *proofreader display* shows a previous version of the text and superimposes stylized markings, functionally similar to a proofreader's blue-pencillings, to indicate the changes transforming this version into the version normal display would show.

Since an entire text does not in general fit on the display screen, the *spatial review* facility allows a user to scroll through his text until that particular portion

is displayed which he wants to examine. Random access to locations in a text, as contrasted with the sequential nature of spatial review, is provided by allowing a user to associate an identifier with a specific location in his text. When subsequently requested, the system displays the text starting at the location associated with a selected identifier.

Because each successive editing change stored in the system defines an incrementally different version of a text, the aggregate of information available is sufficient to recreate the historical sequence of versions corresponding to the successive transformations of the original text. The system provides a *chronological review* facility by allowing a writer to request that any particular version from this historical sequence be displayed, using either display mode. In proofreader display mode, another feature is the ability to show the transformation of any selected version into any later version. Whenever an editor chooses to examine the sequential transformation of his text from an earlier version into a more recent version, he can indicate whether or not the system is to confine itself to the chronological changes in a particular section or whether it is to jump from one location to another in displaying the chronologically successive changes to the entire text. If individual changes are being chronologically reviewed in the latter manner and an editing change involving multiple text fields, such as a move (text deleted in one location and copied elsewhere), is encountered, the system will by default show the spatial area containing the deleted text associated with that change. If the user desires to see the spatial area containing the other field, he can issue a simple system command.

In order to allow an author to *review by editor* the changes made in a text, the system can filter out all editing changes which have not been made by a selected editor, and display in either proofreader or normal display mode only those editing changes or versions which that editor has produced.

DISPLAY AND DATA STRUCTURE FACILITIES

As suggested above, a particular display may contain a substantial amount of information, especially in proofreader display mode. The effort to prevent a display from becoming congested and thus preserve its legibility resulted in the provision of several system facilities dealing with this area.

First, design of the display modes provided by the system had to take into account the limitations imposed by the large amount of information to be dis-

played and the requirements of the graphics terminal employed. Consequently, the design emphasis was on creating a display format which would be as straightforward and canonical as possible. A complete description of the display modes appears in Appendix I. Briefly, for proofreader display, the display screen is divided into two major sections, with one section displaying the text with embedded proofreader marks, and the other containing descriptive information identifying the changes indicated by the proofreader marks (including when and by whom the changes were made).

Second, the system is designed to associate user-specified priority levels with each editing change. In preparing a proofreader display, only those changes with a priority greater than a chosen limit are displayed with the stylized proofreader markings. Remaining changes are incorporated into the displayed text but are not separately indicated. This edit priority facility is designed to prevent the display from becoming congested with proofreader-marked and annotated editing changes of a minor nature (spelling corrections, for example). Otherwise, not only might the significant edits tend to be eclipsed on the display screen by a mass of minor ones, but the system itself might have difficulty in creating the display since the high density of edits could tax the capability of the display screen to simultaneously show both the proofreader-marked text and the descriptive information associated with each change.

Another system capability which reduces the number of edits to be displayed is the ability to specify that particular editing changes are to physically alter the text without being traced. Since no information pertaining to physically executed changes is stored in the system, such changes can never be reviewed or modified. Physical execution of editing changes is thus particularly well suited for minor corrections which do not warrant the retention of tracing information.

Physical execution in a single edition text not only contributes to display clarity, but reduces system overhead as well by eliminating the necessity of dealing with the internally stored editing information. If the evolutionary development of a text has become static after a period of time and the traced information associated with each editing change is no longer needed, the user may specify that all traced edits (or all editing changes up to some particular one) be physically executed. Additionally, a user can request that a copy of the current internal data structure (comprised of the text and its evolutionary information) be placed into archival storage before physical execution takes place. If the need to reference information concerning earlier editing changes ever arose again, the user could call this file into the system and examine it as before. In

effect, this procedure produces a series of files similar to a sequence of hard copy drafts, while retaining the evolutionary information intact and readily accessible.

ACTIVE REVIEW

The active review facilities provide means for a user to modify the current state of his text by altering the effect of prior editing changes. The three principal types of modification are nullification of a past editing change (in which case the current text is altered to appear as though the nullified editing change had never occurred), reacceptance of such a nullified edit, or physical execution of a prior change, whether active or nullified. With active review, an author can reject and thereby nullify the effect of an editing change which he later determines to be counterproductive or which, for example, has inadvertently deleted a portion of his text. Consequently, no ideas can ever be permanently lost from a manuscript unless a writer specifically requests that particular edits be physically executed.

With the addition of these facilities, an author always has five possible actions available to him for dealing with his text. He can make a *new* editing change and have it either traced or physically executed, or he can nullify, reaccept, or physically execute a *prior* change. In order to keep nullified editing changes accessible for possible reacceptance, a third display mode, *full display*, is provided which has the same form as proof-reader display, but which identifies nullified edits as well as currently active editing changes. Since a writer frequently deals with a number of editing changes which logically fall into particular groups or categories (all changes in a section of text, all changes in tense, etc.), the system provides the ability to group an arbitrary number of editing changes so that active review modification will be applied to each edit specified in the group when the group itself is referenced. Although new editing changes must always be performed on the latest (or current) version of text, modification of any past editing change or a group of editing changes can be done without regard to the chronological order in which the edits were originally performed.

Because active review provides for modification of editing changes already made, the fact that edits are not necessarily independent of each other must be taken into account when the system performs active review modification of any particular edit. Any pair of editing changes in a text may be either spatially unrelated or overlapping. If two edits are spatially unrelated, then active review or physical execution of either has no effect on the other. If not, the edits are implicitly related because chronologically later editing

changes depend on the context established by earlier editing changes. Changing an antecedent edit by active review thus has an impact on spatially related later edits. If, for example, a number of relatively minor substitutions were made to a section of text inserted in a previous version, a user would probably want all of these substitutions to be nullified if the enclosing insert was nullified. However, if the changes were of a more substantial nature, possibly substituting whole paragraphs for already existing text, the user might want this material to be retained even if the enclosing insert were to be nullified. The coupling mechanism provided by the system to control the interaction between spatially overlapping editing changes gives the user the ability to choose either of these coupling arrangements, as appropriate, for any particular set of related edits.

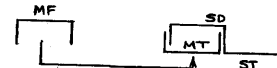
The coupling mechanism employs two coupling fields for each editing change. The *emanate scope* value associated with an edit specifies whether or not modification of this edit by active review will be allowed to affect any spatially overlapping edits. Conversely, the *receive scope* value associated with an edit specifies whether text associated with that edit itself is to be changed when active review modification of an overlapping edit is made. In order for any modification of the text associated with an overlapping edit to take place, both of the following conditions must be satisfied: the edit to be modified must emanate scope and the overlapping edit must receive scope. An example of how edit coupling

The following sequence of changes is made to a section of text:

- 1 Text is moved from "MF" to "MT"
- 2 "SI" is substituted for "SD"
- 3 The move (1) is nullified.

If the coupling between the move and the substitute specifies that the overlapping substitute is to be subjected to the same modification as the move (positive coupling), then nullification of the move will cause the substitute to be nullified. Otherwise, the material inserted by the substitute will remain as part of the current text (negative coupling).

SCHEMATIC:



EXAMPLE:

- CHANGE 1 scenic view in the park
 CHANGE 2 view in the scenic park
 CHANGE 3: view in the baseball park
 POS. COUPLING: scenic view in the park
 NEG. COUPLING: scenic view in the baseball park

Figure 1—Edit coupling example

can be employed is given in Figure 1. The effect of physical execution on overlapping edits coupled so that modification of one is to affect the other should particularly be noted. When an editing change is physically executed, overlapping portions of spatially related edits coupled in this manner to that change are likewise physically executed. Consequently, because physical execution of an edit results in the loss of evolutionary information which cannot be reconstructed, the user must be especially certain that all edits overlapping an edit to be physically executed have the intended coupling values. The ability to interrogate and/or alter both the coupling specification fields of an edit and its priority level is provided. User modifiable defaults for edit priority, coupling field values, and other parameters associated with new, traced editing changes are provided by the system so that a user need supply only a minimum of information when making any particular change.

The concept of active review may be better understood by considering the following model. The text initially input into the system prior to editing defines a base text relative to which all future editing changes are made. Whenever a user specifies that an edit is to be performed, two modifications to the internal data structure are made by the system. The editor's initials, the date, and any annotation explaining the change are stored in that part of the data structure reflecting the chronology of editing changes. All information concerning the scope of each editing change and the operation performed, along with any new material added to the text or moved within it, is incorporated inline into the base text. The chronological components of all editing changes can be thought of as a chronologically ordered set of editing information, to be termed an *edit vector*. The active review facility provides the ability, through edit rejection, to selectively eliminate edits from an edit vector. By performing new editing changes or re-accepting nullified changes, additional edits are included in an edit vector. In this way, the edit vector becomes a selector on the chronologically ordered set of all editing changes, indicating which ones are active (non-nullified) and thus take part in defining the current text version.

In terms of this model, the way in which a version of the text is prepared for display is indicated in Figure 2. To create a proofreader display, the display processor uses the information contained in both the edit vector and the base text to display a version with proofreader marks for all active edits above a user specified priority level. The edit vector acts as a selector by allowing changes to be incorporated into the version of text displayed only if they are specified in the edit vector. It is important to note that *only* the spatial portion of the

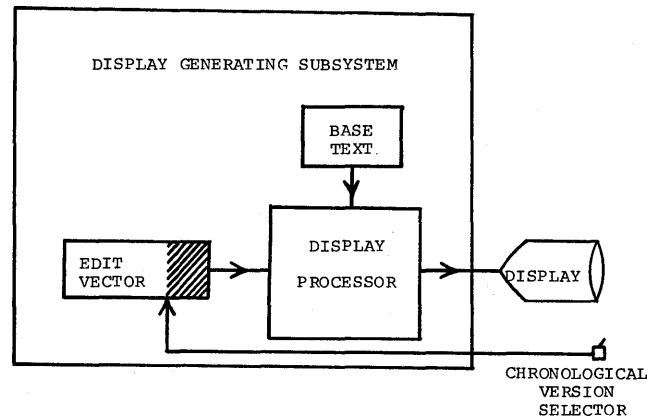


Figure 2—Display generating subsystem

text to be immediately displayed needs to be processed in this manner.

Chronological review, in this light, consists merely of truncating the edit vector at some point, implicitly nullifying all succeeding editing changes. Review by editor is accomplished by implicitly nullifying all changes in an edit vector which have not been made by the selected editor. In this case, only versions of the text created as a result of this editor's changes may be displayed, and only changes made by this editor will be shown in the proofreader display mode. Spatial review, while utilizing an edit vector to define the version of text to be displayed, neither explicitly nor implicitly affects an edit vector.

MULTIPLE EDITIONS

Another major facility provided in this system is the ability to define multiple, distinct text editions which utilize a single, common text data structure. In the above model, this corresponds to the maintenance of multiple edit vectors. This multiple edition facility is particularly well suited for use by a group of collaborating authors. Each author can maintain his personal edition based on the common data structure, selecting editing changes for incorporation into his edition via active review and adding new changes to reflect his particular contribution and style (with or without direct consultation with his colleagues). A composite document can later be produced through comparison of the individual editions and selection of those editing changes representing the best content and phraseology of the group (subject to final editing touches). Alternatively, a single author can use this feature to maintain separate editions of a text which are substantially similar but which differ in presentation to meet the needs of different audiences.

Besides employing editing functions and the facilities of active review to alter a particular edition, the multiple edition facility allows a writer to create a new edition by copying another edition (including its text and evolutionary information). Whenever an editor initiates additional editing changes to a particular edition after multiple editions have been created, he must specify directly or by system default which edition(s) are to be updated (i.e., to which edit vector(s) this edit is to be appended). Each new editing operation is automatically nullified in those editions not specified to receive the change.

Whenever a text has more than one edition defined on it, physical execution of editing changes is performed in an altered manner. In order to comply with the two requirements that all editions be based on a common data structure and that active review changes in one edition not affect any other edition, virtual rather than physical execution is employed. Furthermore, virtual execution allows all changes initiated in a multiple edition environment to be accessible in all editions. As far as a user can tell, the results of virtual execution and physical execution are the same in that editing changes will never again be displayed with proofreader markings and may never be further modified in the affected edition. The difference is that evolutionary information associated with virtually executed changes must be retained even though each such edit appears to be physically executed. Virtual rather than physical execution is always performed when the physical execution function is specified and more than one edition of a text exists. New editing changes virtually executed in specified editions are treated as nullified editing changes in all other editions. In this manner, provision is made for later acceptance of such changes into any of the other editions. When an already existing editing change is physically executed, no modification is made to this change in other editions.

Except as noted above, all information concerning each edit is stored by the system in such a manner that changes performed by an editor in one edition do not appear in any other edition. In particular, an edit can have a different priority or different scope coupling parameters in each edition. A further feature of the system is the ability to spin off any edition as the base text for a separate data structure, either with information concerning all but virtually implemented editing changes intact or with physical implementation of all edits in accordance with the edit vector for that edition.

In comparing separate editions, it is convenient for a writer to have some method for associating locations in one edition with those in another. The system accomplishes this by providing a spatial coupling facility which allows a user to associate a location in each of

several editions with a single identifier. After establishing such locations, a user examining text near a coupling point in one edition can employ the coupling to examine the material at the corresponding location in another edition. Such coupling designated by the user is in addition to the natural coupling inherent in the data structure—unless specified otherwise, whenever the user switches editions, he automatically sees the text in the same general spatial area.

CONCLUSION

It is the principal objective of this particular computerized writing system to present in one implementation many of the separate advantages favoring other kinds of editing systems. By providing writers with this new, generalized writing system characterized by ease of editing and both review and use of evolutionary information, it is hoped that writers' abilities to create texts may be significantly augmented.

Any implementation realizing these facilities, by the nature of the system's massive information storage and processing requirements, places a heavy demand on computer system resources. Although further advances will be made in computer hardware capabilities and in the creation of display presentations which are more human factored, the current experimental implementation of this writing system is significant because it will allow determination of the extent to which the ability to passively review and actively modify evolutionary information does augment the writing process. It also provides for establishment of criteria against which the effectiveness of other means for displaying a text's evolution can be judged.

ACKNOWLEDGMENTS

The authors wish to express their appreciation to William P. Braden, John V. Guttag, and Daniel E. Stein for their contributions to the design of the computer assisted writing system described herein.

REFERENCES

- 1 D C ENGELBART and W K ENGLISH
A research center for augmenting human intellect
Proceedings AFIPS 1968 Fall Joint Computer Conference
Part 1 1968
- 2 S CARMODY W GROSS T NELSON D RICE
A VAN DAM
A Hypertext Editing System for the/360
Proceedings 2nd Annual Conference Computer Graphics
University of Illinois Urbana Illinois 1969

APPENDIX I

DISPLAY FORMATS

A user can choose to view his text in any of three modes of display—normal, proofreader, or full display. Normal display consists simply of a text area in which an updated version of the text is displayed in double spaced ragged-right lines, and a prompt area at the bottom of the screen in which system messages for the user are displayed.

The proofreader display (Figure A1) consists of a text area, a descriptor area, and a prompt area. The text for the chronological version of a particular edition is comprised of not only text presented by normal display but also text associated with active (non-nullified) deletions; proofreader marks are superimposed to identify editing changes. A line is drawn through all deleted text in order to distinguish it from current text, and vertical lines are embedded in the text to delimit the spatial scope of each editing change. When the vertical line denotes the beginning of a scope, it rises to meet a horizontal line above the text line; when it denotes the close of an edit's scope, it drops to meet a similar horizontal below the line of text.

The descriptor area provides detailed information about the editing changes displayed and is comprised of marginal areas on both sides of the text, a tag block, an annotation block, and a prompt area.

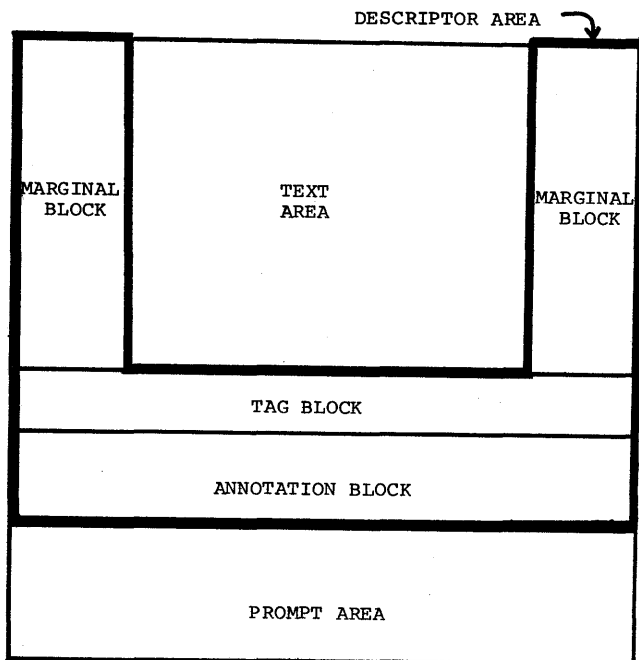


Figure A1—Proofreader and full display format

1. SEE TAG 1	The principal le American source of	SEE TAG 2
2. 3I;4SI;4SD	the ideas of used by the professional	4SI;4SD,3I
3. 5MD	reformers was the 1830's Military	5MD
4. 6SI;6SD	Enlightenment Reforms in England.	6SI;6SD
.	.	.
.	.	.
.	.	.
TAGS:		
1. 1SI;1SD;2SI;2SD		
2. 1SI;1SD;2SI;2SD		
ANNOTATION:		
1. SUBSTITUTE; 170; WDE; 4/7/71; SI: 1/1; SD: 1/1		
2. SUBSTITUTE; 213; WAP; 4/8/71; SI: 1/1; SD: 1/1 *		
3. INSERT; 73; WAP; 1/5/71; 2/2		
4. SUBSTITUTE; 215; WDE; 4/8/71; SI: 2/2; SD: 2/2 *		
5. MOVE; 108; AVD; 2/1/71; MI: 6/6; MD: 3/3 *		
6. SUBSTITUTE; 234; AVD; 5/4/71; SI: 4/4; SD: 4/4		
FUNCTION IGNORED: NEW CHANGES MUST BE MADE TO CURRENT VERSION		

Figure A2—Sample proofreader display

and an annotation block. In the left margin, editing changes beginning their scope on a line are identified by short codes indicating their edit type. Similarly, the closing scopes of editing changes are identified in the right margin. The order in which these codes are listed corresponds to the order in which the edit scope delimiters occur on each line. If there is insufficient space to list a given line's codes in the appropriate margin area, this information is placed in the tag block and a reference to this tag is made in the margin.

The annotation block contains a fuller description of each change, including the edit number, the editor's initials, the date of the change, and an indication where each end of its scope lies. Figure A2 indicates how a representative proofreader display appears.

Full display presents the text and all editing changes, whether nullified or active, which are included in the version of the particular text edition displayed. In order to distinguish between accepted and nullified edits, text within the scope of the former is displayed in small letters whereas that of the latter is shown in capital letters. This display, as contrasted with the proofreader display, uses lines drawn through text to identify not only text deleted by active edits, but also text inserted by nullified edits.

The following example illustrates the manner in which an edited line of text would be displayed in each of the three display modes. For the sake of clarity, the example has been chosen so that the edits have no overlapping scopes. The initial text to be edited is the phrase "hard copy editing". The first change is to insert "text"

```

NORMAL DISPLAY:      evolutionary editing
PROOFREADER DISPLAY: hard copyevolutionary editing
FULL DISPLAY:       hard copyevolutionaryTEXT EDITING

```

Figure A3—Display mode example

after “copy”, resulting in “hard copy text editing”. Then “evolutionary” is inserted after “copy” producing “hard copyevolutionary text editing”. “Hard copy” is then deleted, resulting in “evolutionary text editing”. Finally “editing” is deleted, resulting in “evolutionary text”. If upon reviewing his edits the writer accepts his second and third edits while rejecting the first and fourth edits, the edited line would appear on the three displays as shown in Figure A3.

APPENDIX II

INTERNALS OF THE SYSTEM

A text's internal data structure consists of two major and a number of ancillary areas, all of which are arbitrary length segments and program pageable to disk.

The two main areas are the text area, which contains a linear string of text with embedded edit codes defining the nature and scope of each editing change, and the edit area, which gives the location in the text of each edit and contains additional information concerning the execution status and priority of each traced edit in each text edition.

Every time a traced editing change is performed, the text pertaining to that change is bracketed inline by a pair of edit codes and an entry is made in the edit area. The correspondence between edit codes and the related edit area entry is maintained by the use of edit numbers. The edit's start code, besides delimiting the scope of a change, contains the bulk of the information describing an edit which is required by the display processor. This information consists of the edit number, an index into the table containing any annotative information pertaining to the edit, the type of editing change, and the priority, nullification status, virtual execution status, and the emanate and receive scope values for the edit in each edition. The edit's end code, which contains only the edit number, merely delimits the end of the scope of a change. Each page in the text area has a header which lists those edits whose scope overlaps from the previous page. Consequently, the display processor must only go back to the beginning of a page in order to determine text to be displayed in any spatial area.

Planning computer services for a complex environment

by JOHN E. AUSTIN

Harvard University
Cambridge, Massachusetts

INTRODUCTION

A responsibility that is being faced by more and more corporations, research laboratories, universities, government agencies, and other large complex organizations is that of providing effective computer services from a variety of sources to serve a multiplicity of user needs. At Harvard University this problem has been faced for a number of years and various planning processes have been developed to cope with it. In the 1970-71 academic year a new office was created, the Office for Information Technology, with planning and coordination of computer services as its first priority.

The University at that time had an IBM 360/65 in its Computing Center providing batch processing, timesharing, and process control services through a central batching room and a number of remote job entry stations. There was also a 360/30 operated by the Comptroller's Office. In addition to these two facilities, there were a number of smaller computers located in the various departments, and considerable use of time-sharing services purchased from commercial vendors. At the time the Office for Information Technology was created the Computing Center was incurring deficits because of a downturn in research funding and because of shifts of interest to other forms of computing services. Because of the uncertainty about the future, the Office was asked to analyze the usage of computing in the University and to propose alternatives for services.

This paper summarizes part of the analysis and several of the alternatives that were proposed. The actual choices and implementation processes involved considerable negotiation. But for non-technical administrators of any complex endeavor to make a decision that has a number of technical ramifications, they must have a point of departure and this paper represents just that.

The choices, as they would be for the corporation, research laboratory or government agency, involved neighboring institutions, which in Harvard's case is the

Massachusetts Institute of Technology (MIT). They also involved commercial services. The criteria for selection, in the last analysis, were a combination of cost-effectiveness to the University and flexibility for the users.

CLASSES OF COMPUTER USE

We found it convenient to divide uses of computers at Harvard into six basic classes: User written program compilations/executions, Use of package programs, Large scientific production, Administrative production, On-line data collection and process control, and On-line interaction. The adequacy of these descriptions can be argued, but they show up as groups in the statistics derived from the 360/65 at the Harvard Computing Center, as well as in discussions with faculty and administrative groups throughout the University. An individual can be a member of different classes at different times in that, for example, an administrative or scientific production job requires program compilation at some previous point in time.

The following sections will discuss characteristics and volumes of these types and outline alternative sources of services.

User written program compilations/executions

The largest user class of batch processing at the Computing Center in numbers of jobs per day is represented by the person who has written a FORTRAN program and wishes to have it compiled and, frequently, executed. He typically requires less than 150k bytes of core and during a typical week he had an average turnaround time of 23 minutes. This class accounts for 36 percent of the number of jobs at the Computing Center.

It is more difficult to measure the extent to which this use is made of the commercial timesharing system.

At the Harvard Business School (HBS) it is probably less than 10 percent. In the Faculty of Arts & Sciences (FAS) it may be as high as 90 percent. The reasons are pedagogical: the FAS students are learning how to program, the HBS students are learning how to analyze management problems.

Some members of this user class also have available to them certain other facilities. There are the IBM 1620 computers available to users at the School of Public Health, in the Biophysics and Chemistry departments. Members of the Physics Department can use the XDS Sigma 7 located there.

The needs of this user class can be generally described as follows:

Rapid turnaround time on an unscheduled basis is frequently important. If the individual is working on a project, he usually wishes to get on with it as quickly as possible. In the case of students, it may be vital to the completion of an assignment.

Means of providing this service can depend on the size and computational requirement of the program. If it is small and requires small amounts of core and CPU time, timesharing offers the obvious advantage of being able to complete the task, including several iterations, in one session. If the program is large in the number of coded instructions or in the amount of data to be processed or in the amount of output to be produced, then batch processing can be cheaper. In some cases a combination of terminal access to a system with a deferred batch execution is the best method.

Service alternatives

This user class could be served in several ways. For the batch user whose mode of entry is punched cards and whose principal output is paper listings, a relatively convenient card reader/printer station is sufficient, the location of the central processor not being important.

For many users it is essential to have a consulting service available at the reader/printer station to help them with problems they encounter in attempting to run their programs. The range of this service will be that currently covered by the programming assistants at the Computing Center. In addition, it is important that a source of information and advice on all features of the system be available to the user.

For the timesharing user in this class whose mode of entry is typing on a keyboard in his own work place, ease of remote access to the system, reliability, and fairly fast response are his main concerns. He would like to be able to depend on the time when he can get on

the system, to know that it will remain operational, and know that he can get his work accomplished within a reasonable time.

Use of package programs

The next largest user class, and one that is growing, is the person who comes to the computer to use a program that has been prepared by someone else. This may be a statistical program like Datatext or SPSS, or it may be a structured presentation of a large data base like the King Charles County Case or the Management Game at the Business School. The user in this class may bring his own data to be processed by the package program, or he may be given a decision-making aid in which both the logic and data are part of the package. This user is seeking a service that is more than simple computation. The computer (and its associated software) is responding to the user in terms related to his kind of work.

At the Computing Center this currently represents about 12 percent of the jobs. On the timesharing system this class represents most of the HBS usage (90 percent) and a small but growing amount of FAS usage (10 percent).

The service needs of package program users vary greatly depending on the size, computational requirements and intended use of the package. For some packages used mainly by a local subgroup of users these functions will be performed locally but for packages of more general interest it may be determined that the package is of such universal appeal as to be managed centrally.

Service alternatives

By and large this usage class is not interested in proximity to the central processor. Since the results are the main concern, and since packages are frequently obtained from many different service sources, a rule of thumb would be that cost-effectiveness to the user should dictate where the computing is done.

Program packages are in one of the fastest growing segments of the computer market. It is also one of the areas in which universities make major contributions and can profitably share their results. Because of the diversity of systems used throughout the academic and commercial worlds, it is important to the potential package program user that he not be constrained from obtaining this service because it will not run on the system he is required to use.

It should be noted that the requirements for package development are different from those of package use.

There is a general fear that if service requirements are met solely from outside sources, the resulting inconveniences will inhibit experimentation and development of new packages.

Large scientific production

This usage class may overlap with the package program user to some extent. Its distinguishing characteristics are that it uses large amounts of core and/or CPU time relative to input/output. The programs may be lengthy and by running again and again with new data, solve complex problems in chemistry, physics, astronomy, and other sciences.

This usage class does not represent a large number of jobs at the Computing Center (about 4 percent), but the demands on the resources are such that the costs incurred to serve them are high; and their portion of income to the Center is also significant. Of the approximately 320 active customers of the Center in a recent month, twenty of them provide 50 percent of the income.

Historically it was the large, government-supported scientific project that was the high priority user of the Computing Center. Other activities are beginning to catch up in volume and dollars with this usage class, but government-supported work still tends to get considerable attention because of the general overhead component of the research contract.

The needs of these users are:

Large resources. They will use high speed processing and large quantities of core in abundance. Programs will tend to conform to the largest dimensions of the system, whatever they are.

Reasonable rates. Most of these users have access to government-financed computing centers. When the local rate is too high, the differential will overcome the inconveniences of going elsewhere to get the work done. Members of this class will do as much computing as they can get money for.

Reasonably convenient access to the machine. Most of these users have a high technical skill, need very little help from the staff, would like to use the computer as they do any lab equipment.

Service alternatives

This class uses batch processing almost exclusively because large CPU/large core jobs do not blend well with a multi-user environment. Satisfactory service could be obtained from a number of sources including a Harvard-operated center, MIT, or a government-

financed center. To the extent that this class is directed to other centers there is the risk that they will feel unsupported by the University.

Administrative production

One of the best understood and most visible classes of computer usage in a university is that of administration. These users have several things in common: they all face deadlines which they must meet; they can schedule most of their work long in advance; and they have considerably more input/output relative to computation than any other type of user. In addition, there is considerable need for security of files and for the operational environment to be protected against unauthorized access.

At the present time about half of the Harvard administrative jobs are run at the Comptroller's Data Processing Center on the 360/30 and half at the Computing Center. The Comptroller's workload is mostly his own with some work being done for the Personnel Office, Widener Library, and others. The Computing Center jobs are those of the Printing Office, the Press, the Medical School, the Development Office, the Admissions Office, the Law School, the Division of Engineering & Applied Physics, Widener Library, the Registrar of the Faculty of Arts & Sciences, and the Business School.

The needs of the administrative production class are:

Dependable scheduling. They must be able to rely on completion of their jobs at the appointed time.

Security. Files and output must be protected against unauthorized access.

Data control. More attention to operational considerations is required of administrative work.

This could be a growing area of interest at Harvard. The possibility of having a comprehensive information source will enable the Deans and the budget officers of the various departments to plan better and to monitor and to analyze their operations throughout time.

Service alternatives

As a user of batch processing, the administrative production class requires a machine with a high level of disk, tape, and printer capacity to do the work. Technically there is no compelling reason to have the machine located in or near the administrative offices. In most places it happens to be so located and there would probably be a strong desire to have at least a

high speed card reader/printer station in the administration building for Harvard administrative production.

It is important to recognize that this work requires the mounting and storage of many reels of tape and disk packs under tightly scheduled conditions. Operator error can be both painful and costly, as would theft or other forms of abuse. Any combined center doing administrative work, whether Harvard-operated or elsewhere, would have to make very careful provision for this aspect of the operation.

On-line data collection and process control

There is a class of computing at Harvard that is much less visible than the others but is certainly a sizable one. There are a number of laboratories in which a part of the instrumentation consists of a small computer to collect experimental data or control experimental processes. In some cases these same computers analyze and display the data with plotters, printers or CRT displays.

The most important current consideration in this class is the service provided by the Computing Center to the Cambridge Electron Accelerator. The commitment made by the CEA was 10 percent of the total dollar amount pledged to support the Center this year. The arrangement is that when the accelerator is actively conducting its colliding beam experiments, the 360/65 acts as a high speed data collector at the end of a cable directly connected to the accelerator. The data is processed in 100k of core in the 360/65 and displayed on units back at the CEA facility. The important elements in the relationship are the direct connection over a high speed cable, continuous on-line operation for long periods (of up to two weeks), the cycle time of the 360/65 CPU and the 100k of core.

It is essential to the CEA that a computer with a speed of at least that of a 360/50 and 100k of core be available at the end of a high speed cable for the life of the colliding beam project. The service alternatives are quite limited from a practical standpoint, because the reprogramming required to run on another machine would be added delay and expense.

On-line interaction

Interactive timesharing is the class of computer use that has had the most growth, the most problems, and is in the opinion of many people the most important class for the future. It is not an exclusive class in that program compilation and use of program packages are the primary timesharing activities. Its distinctive

feature is the opportunity for the user to interact conversationally (as they say) with a program. Because of that, and because the service sources have to be thought of in different terms, it tends to be considered as a separate kind of computing.

Experience with commercial timesharing service at Harvard has been that the demands have exceeded the supply very early in the life of any contract. There have been many difficulties in getting the level of service that we thought we were contracting for, but in spite of those difficulties many, many thousands of terminal connect hours have been used. Members of the several faculties predict that the demand will increase by at least 50 percent next year.

The Computing Center announced its CALL/360 service early in February, but there has not been enough time to tell whether that service will be used to its capacity. Continuance of CALL/360 beyond the Spring Term depends on user response and billings.

The needs of individual timesharing users are those previously mentioned: reliability and responsiveness. In the University community there are several larger needs:

Capacity. As demands grow, they continually exceed the ability of any one system to satisfy them. When several sources are used, there is a problem of storing programs and files on multiple systems, and if they are different, there is the problem of incompatibility.

Languages and other software subsystems. The Harvard community has not only large demands but a diversity of needs for different computer languages and package programs. These are not always available on one system, and one system cannot always operate effectively if it tries to offer services at too many levels.

Variety. Compared with batch processing, timesharing can be a much more pleasing mode of operation. For this reason classes of users now content with batch processing may convert to timesharing, and it will be necessary to partition the class of timesharing users into finer categories.

These categories are the same as the ones above, namely:

User program compilations/executions. It is extremely convenient (to the point of inducing carelessness) to write programs in the timesharing mode because editing, testing, correcting can follow each other rapidly.

Package programs. The possibility of having a library of commonly used programs should remove

the need of having small computers such as the IBM 1620s since many of the programs are small and the users don't need a computer all the time. Many of the data analysis packages are more useful when run in timesharing mode. These will also require setting up, maintaining, and retrieving from large data bases.

Large scientific production. Although users will not sit and wait for such jobs to be done, the initiation and control of such jobs may be done by remote access to the computer.

Administrative production. It is convenient to update the information in files and to do editing of such files on-line. The capability of doing on-line analyses by timesharing is then available.

Service alternatives

There is such a diverse set of needs for timesharing that one source is probably not the best solution to the problem.

A contract with a commercial service corporation is one alternative for one type of service. For next year there are various other alternatives or additions possible. We could replace the commercial service with another timesharing vendor. We could try to renegotiate the present contract at a lower level and supplement it from another vendor. We could install one or more small basic one-language timesharing machines like the HP-2000 for the use of beginning programming students. We could offer CALL/360 on a more extended basis if the trial use on the 360/65 proves effective. There is also the possibility of some service available from MIT on Multics.

In any case, more timesharing service than is now available will be required for next year and in the years to come. Furthermore, we predict that other usage classes will tend to convert gradually to on-line systems as such systems acquire greater capacity, become more reliable and evolve into networks.

ALTERNATIVES FOR PROVIDING COMPUTER SERVICES

The minimum risk alternative

A general review of all six classes of computer use points up the two fixed obligations for Harvard-operated computers: the service to the CEA and the administrative production, which at this point is mainly the work for the Comptroller. There has to be a computer with the speed of a 360/50 at the end of the cable stemming

from the CEA and a 360/30 or its equivalent to serve the administration with input and output under administrative control. These obligations are both technically and organizationally based. All other services could be procured from other sources.

During the month of February this possibility was explored with two batch processing computer service sources: MIT and a commercial vendor. The user classes under consideration were user written program compilations/executions, use of package programs, and large scientific production.

In our analysis of rates we found that the jobs run by the Harvard Computing Center in January would have cost about .9 times as much at MIT and 1.3 times as much at the commercial vendor. We tried to consider the logistical problems in getting that volume of work done at these two places, and found that the work being submitted from IBM 2780 RJE stations could be processed much as they are now with a few changes in the job control cards. The large problem would be handling the work now submitted at the Computing Center batching room. In all likelihood we would have to provide a very high speed I/O terminal (on the order of a 360/25) in addition to regular courier service. We would also have to provide assistance to users as we do now and have at least one systems programmer who knew the other system thoroughly.

At the same time, the Office for Information Technology would help find other sources of both batch processing and timesharing services, as is done now, for special needs. If large scientific production users could get a better deal at the AEC financed center, arrangements for a remote job entry station could be made for them.

This alternative has the least financial risk to the University in that it involves the lowest fixed-cost arrangement that can be made. It also carries with it a more complicated management problem in making all these services effective (as has been the case with the timesharing service contract this year).

The minimum change alternative

Given the utilization rates on the 360/65, it is the opinion of the Computing Center staff that the University needs a machine of that size to do the work that needs to be done. Replacement of the 360/65 by the recently announced IBM 370/155 has a number of unquestionable advantages.

The hardware costs for an equivalent machine are lower.

The design of the 370/155 and its peripherals

provide for a greatly enhanced file handling capability.

The 370/155 could emulate the 360/30 DOS operation run by the Comptroller, and therefore, eliminate the need to either do a quick conversion of his system or to keep the 360/30 beyond this next summer. The conversion to OS/360 would continue, but at a more thoughtful pace.

All of the present users of the 360/65 at the Computing Center could continue to get services with no disruption whatsoever, including the service provided to the CEA.

This alternative has at least two possible versions:

A 370/155 configured to meet the batch processing load of a 360/65 only could be installed in the Computing Center building. This would be a simple replacement of the 360/65.

The only major technical problems to be overcome in this case are those connected with emulation of the 360/30 prior to the Comptroller's conversion to OS. Many of those DOS programs require operator intervention which complicates life in a multi-programming environment.

The second version of this alternative would be to put the 370/155 on the fifth floor of the administration building in the area now occupied by the Comptroller's 360/30. On that same floor there would be space for a separate I/O area for the Comptroller. There would then be a fast I/O station in the batching room at the Computing Center building to handle the work submitted there.

This version has the advantage of giving greater security to the hardware and to the files which would be an improvement from an administrative point of view. It has the disadvantage of having to provide a path for the CEA service over the broad band communication cable from the Computing Center to the administration building.

This alternative, in either of its versions, is a continuation of the Computing Center concept as it exists now but at a lower cost. There would still have to be a procurement of special services—particularly timesharing services—from other sources.

The integrated center alternative

An alternative building on the previous one would be to structure a system that would combine batch and remote batch processing with a timesharing service. The 370/155 is designed to take advantage of multiple channels and high speed disks for both batch and on-line

file handling. With the addition of 500k bytes of high speed core a 50 terminal CALL/360 system and CRBE would be able to run with minimum interference with the batch stream. In about a year IBM's new Time Sharing Option may be running under OS and could provide a significant new form of computer access.

The advantage of this alternative would be a substantial increase in timesharing capability for those who needed only BASIC or FORTRAN at a medium cost. There might also be a number of users who would welcome TSO when it becomes available.

LONG-RANGE IMPLICATIONS

What do these alternatives point to beyond the immediate planning period?

The minimum risk alternative suggests that Harvard wishes to relinquish a large part of its computer services operations business. It also opens the way for a combined center with MIT if that should be desirable for the two institutions.

The risks of this alternative are that control of service quality would be in someone else's hands and this could have repercussions among the many computer users who judge a university on its resources. On the other hand, it is just as likely that such a move could be among the first of many such moves by universities located in urban areas where diverse computer services are locally available. It is no longer true that computing is an exotic activity requiring a research and development environment. While the field is still undergoing great change, there is reason to believe that buying services as needed will prove more beneficial to the educational and research institution than attempting to support them at a very high level internally.

The minimum change alternative suggests that there is a cost-effective-level—to be determined by financial commitments from users—at which some internally provided services can be maintained. Just as we continue to have a Buildings and Grounds Department and a Printing Office, so we should have a computer service center where University needs can be met by resources managed by other University staff. There is an opportunity to influence priorities, to control costs and to dictate what services and service levels will be provided. In addition, the administrator gets the security and control that his files require.

In the years ahead, we would be saying, there is a continuing need for this and it is part of the overall service component of a university.

The integrated center alternative suggests that the services of a single fast processor can handle a variety of computing needs effectively and it may well be demonstrated that it can. The advantages are that one management takes care of everything and that is much less wasteful than having several faculties each doing their own thing.

The costs of getting services from many sources would be higher in some cases and lower in others. The advantages of diversity would have a cost in inconvenience. The maintenance of a resource center is a hedge against fluctuations in the marketplace. Everything has its price.

BIBLIOGRAPHY

- 1 D N FREEMAN J R RAGLAND
Response-efficiency trade-off in a multiple university system
Datamation pp 112-113 March 1970
- 2 F WARREN McFARLAN
Problems in planning the information system
Harvard Business Review pp 75-89 March-April 1971
- 3 CHARLES MOSMANN EINAR STEFFERUD
Campus computing management
Datamation pp 20-23 March 1 1971
- 4 ANTHONY RALSTON
University EDP: Get it all together
Datamation pp 24-26 March 1 1971
- 5 MICHAEL M ROBERTS
A separatist's view of university EDP
Datamation pp 28-30 March 1 1971

A high performance computing system for time critical applications

by T. J. GRACON, R. A. NOLBY and F. J. SANSOM

Control Data Corporation
Sunnyvale, California

INTRODUCTION

The increasing complexity of current time critical computer applications has generated requirements for large scale, general purpose, digital computers in real-time systems. Yet, few real-time applications can singly provide the economic support for such a major system. This high computational capability-reasonable cost/study need has led to the development of systems that are capable of running two or more time critical jobs concurrently, in addition to local batch processing, remote batch, communications, and interactive graphics.

This paper discusses recent design improvements* in such systems. The new design handles the two most important system tasks in a multiprogramming real-time system (CPU time scheduling, and real-time data interface) through a hardware real-time monitor which schedules tasks (interrupts) using a relative urgency algorithm and an extension of the central memory of the system to provide for data input/output. Also provided is the means to guarantee job integrity so that up to 15 time critical jobs can run concurrently.

The system is currently being implemented for the Naval Air Development Center, Johnsville, Pennsylvania, for use in supporting research and development activities in the field of naval aviation. The computer application areas include real-time man-machine simulations, acoustic research signal processing, a quick-response capability for Southeast Asia problems, direct support of operational systems, a batch processing and graphic capability for general scientific and engineering problems, management information, and computerized NIF accounting.

* While some of the concepts providing the philosophy and theoretical basis for these designs have been previously reported (see Reference 1), this paper reports the first known implementation of them in a system.

SYSTEM DESIGN

Time critical tasks—response

An application is said to be time critical if it demands a response within a fixed time after it has received a stimulus (i.e., interrupt).

The system responds in two ways. It first must sense the interrupt, and capture all data needed to process the task associated with the interrupt. Then it must perform the processing required and have the results available within the required time. In most hybrid systems, devices such as sample/hold units and data buffers are provided in the interface to automatically store the current values of the needed variables at the time of the interrupt. The system maintains the responsibility for sensing the interrupt and performing the required computation on this captured data within the time tolerance allowed.

The extension to a multiple interrupt job is straightforward. A good system will recognize all active interrupts and schedule the required tasks so that *each* task is completed when the interrupting system needs the results.

Note that in a system which guarantees that all interrupts will be processed within the required time, there is no reason for a preferential (priority) treatment of any interrupt. Internally, the system often has to select between conflicting requests while scheduling the tasks, and does so through a time-dependent priority rule evaluation. This CPU scheduling algorithm is hardware implemented in a special unit named the Hardware Real Time Monitor which is discussed later. The same interrupt task may have different priorities depending on the state of all requests at that time. This is an internal procedural matter, and the user has no need to specify priorities between his tasks.

In fact, letting the user specify priorities for interrupt

processing in a multiple job environment causes difficulties. For example, where a priority tree is to be shared among simultaneous users, the users must meet and decide by committee which interrupts are assigned to each job. Inevitably, each job finds itself running in an environment with other jobs that have interrupts of higher priority than its own. The result is of course that a job's successful execution is dependent on the benevolence of the other jobs concurrently in execution. Since the allocation of interrupts between users who are resident simultaneously within the system will vary as jobs enter and leave the system, it becomes impossible to guarantee consistent results from multiple runs of the same program.

To prevent any possible interjob interference, central processor time must be properly allocated. Before a time critical job is allowed to start, the system must guarantee that it can coexist with the time critical jobs presently running. The method used in the NADC system of analyzing this situation is a static scheduler system program which is run prior to the job being allowed into real-time status. This program compares the worst case requirements of the time critical job requesting initiation against the worst case requirements of time critical jobs running in the system. Control card information such as frame time (FT), required compute time (RCT), and the number of analog and digital channels being converted each frame time provide the static scheduler with sufficient information to determine if the requesting job will fit, without conflict, into the system.

If the petitioning time critical job will possibly conflict with other running time critical jobs, it will not be allowed to enter the system. Of course, the system operator has ultimate control over all jobs in the system and can suspend a running job to release time if the facility manager decides to assign a higher priority to the requesting job.

Once the new job is allowed in the system, the task of maintaining job integrity is reduced to a task of monitoring the individual interrupts for violation of the parameters supplied on the job card. The monitoring is done automatically in the Hardware Real Time Monitor.

Time critical tasks—data handling

Traditionally, real world data has been handled in a time critical system in a manner similar to I/O data in a conventional computing system. The data is captured by the A/D conversion gear under the guidance of a device controller and then transmitted through a normal input channel into some buffer from which it is

operated on by the CPU. Output is performed in a similar manner with data loaded into buffers from which it exists on a data channel through a device controller and into the D/A gear.

The approach, while proven workable in a large number of systems, has some drawbacks. It requires that a portion of the systems I/O capability be devoted to, or at least be on short call notice, to the real-time instrumentation, limiting the amount of normal I/O processing that can be done. Normally, the systems I/O facilities were designed for non-real-time data transfer and have inherent design traits such as low bandwidth and a time consuming "activate" requirement which limit their performance in a time critical applications environment.

To alleviate some of these problems, a special I/O port was designed for the NADC system. This unit, called DADIOS for Direct Analog Discrete Input/Output System, exists as an extension of central memory. Every A/D and D/A channel in the system has a one word storage buffer which is directly addressable by the CPU. The data conversion is controlled by a combination of timing and interrupt signals from HRTM and channel addressing capability within DADIOS. The net effect of the system is that real-time data can be transferred into and from the virtual central memory without requiring any CPU time or standard I/O resources. Other capabilities include hardware fix/float conversion and program controllable allocation of pooled instrumentation to jobs. The use and design of DADIOS are discussed later.

SYSTEM OPERATION

The job processing analysis presented in this section begins with a variety of scheduling algorithms, and the aspects of job entry, including control card requirements, job control, and system monitors. Data transfer and any required analog control across the interface during a time critical run is discussed, followed by methods employed in the system under discussion to recognize interrupts and cause the CPU to begin processing the interrupt-specified task.

Interrupt philosophy and scheduling

Before discussing the hardware and software concepts utilized in this system, it is necessary to define "interrupt response" and "scheduling."

In a conventional signal real-time job environment, the response to an interrupt will normally be quoted as the time from interrupt stimuli until CPU action is first initiated (assume interrupt is highest priority). The

first action taken by the CPU is usually the initiation of the data conversion cycle. Even in this environment, this is not the most meaningful measure of interrupt response. Rather, response should measure the period from stimuli until results are obtainable. This response is, then, an "outside world" response which also is a function of CPU speed and the manner of treatment of interrupts within the system. Outside world response requirements are usually dictated by the problem which is generating the interrupt. The important parameter is not how fast the machine can get the CPU working on the highest priority interrupt, but rather the period of time in which the machine will guarantee finishing the interrupt routine for each interrupt specified.

In the multi-real-time job environment, interrupt response is again best specified by the outside world response, that is, from stimuli to completion of results. Since we are dealing with multiple real-time users, levels of interrupt take on a different meaning, for the response requirements of each of the multiple users must be satisfied. A requirement thus exists to devise an algorithm that will schedule interrupts into the CPU. There are two requirements for the execution of this algorithm:

- Outside world response will be guaranteed to meet the requirements specified by the user.
- Absolute integrity between jobs must exist. That is, any attempted CPU overruns of one job may create problems in that job, but must not be allowed to affect another user's CPU allocations in the system.

This algorithm is broken into two portions. The first is termed a static portion (Static Scheduler—non-real-time), and the second is called a dynamic portion (Dynamic Scheduler—real-time). The Static Scheduler ensures that job mixes with potential conflicts in system resources do not coexist in the machine. The schedule is determined in batch mode prior to the job entering real-time status. The scheduling is performed with the worst-case conditions of the algorithm utilized in the Dynamic Scheduler; a worst case fit of the real-time job entering the system is calculated and compared to the worst case conditions of all real-time jobs presently running in the system. If the job fits, it is allowed to proceed into real-time; otherwise, the user is notified that system resources are not available.

To realize the importance of dynamic scheduling it must be viewed as a task of monitoring jobs for violations of specified parameters. The Static Scheduler had determined, prior to job entry into the real-time state, that the job would fit within all restraints of the dynamic scheduling algorithm used, provided the job

abides by the parameters used in statically scheduling it into the system. The specified parameters (RCT, Tolerance, Period) are dynamically monitored by the Dynamic Scheduler and any job which attempts to violate any of these parameters is flagged and aborted for one Period. This action prohibits the job from interfering with any of the others in the system.

When a job is put into real-time, the Dynamic Scheduler takes over the scheduling of interrupts in the system.

Many algorithms for dynamically scheduling the CPU have been devised, and three of the more common methods are presented in the following discussion.

Time slicing

The time slicing algorithm is perhaps the easiest to understand. As indicated by Figure 1, a specified period of CPU time is assigned to specific jobs in a fixed pattern every revolution (R) of the time slice wheel. Interrupts for a specific job have priority only during that job's time slice. For example, if the CPU is active in time slice A, only interrupts associated with job A will be allowed into the CPU. The priority of interrupts within job A will be determined by the priority structure of its hardware or software interrupt tree. When time slice B is reached, all work on interrupts for job B will have priority. Any unused time, of course, will be assigned to batch work.

The time slicing algorithm basically attempts to synchronize several events by slicing all events into many synchronous subevents. A goal in this algorithm

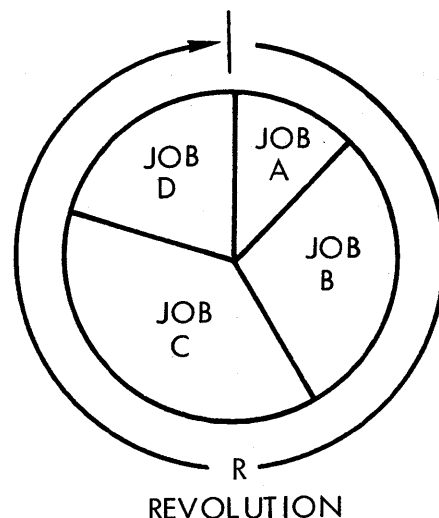


Figure 1—Time slicing algorithm

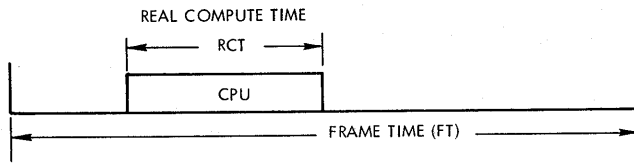


Figure 2—LTTG-JB parameters

is to make the revolution (R) of the time slicer very small, such that response to the multiple users is fast. The inherent system overhead is a practical problem encountered when the system is forced periodically to jump from one job to another. The time slicing algorithm can be easily implemented with either a software or hardware monitor.

Least-time-to-go job basis

The least-time-to-go job basis (LTTG-JB) scheduling algorithm enters a factor of complexity over time slicing into the process of scheduling, but also provides a more efficient scheduling mechanism (it can be illustrated that time slicing is a synchronous subset of LTTG-JB). The two parameters required per job scheduled are as follows (Figure 2):

- Frame Time (FT)—This is the scheduling frame time which is often identical to the problem frame time. However, if multiple interrupts are used in a job, the scheduling frame will usually be equated to the most critical interrupt in the job.
- Real Compute Time (RCT)—The RCT is the total CPU time specified per FT for the servicing of all interrupts which may require service during that FT. Actually, the CPU time will be split into many segments, but the composite of these segments will not be allowed to exceed the RCT specified (see Figure 3).

The above parameters will be supplied by the user

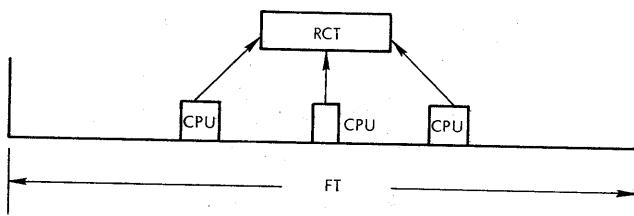


Figure 3—Real compute time

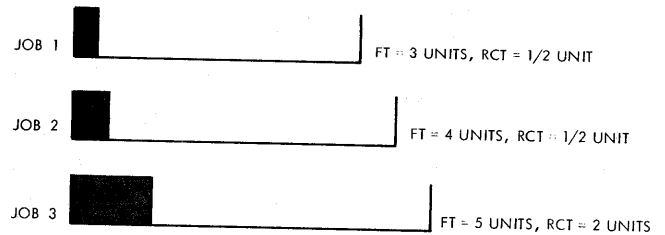


Figure 4—Job specifications

in a FORTRAN call to the Static Scheduler to determine worst case job fit. If the job is allowed into real-time mode, these parameters will then determine which job has priority at a given instant of time by calculating which job has the least-time-to-go before CPU calculations for a particular frame must be complete.

Explanation of this algorithm is best given by illustration. Assume a three job situation where specifications of FT and RCT for each job are as indicated in Figure 4, and each job consists of a single interrupt. As illustrated in Figure 5, job priorities are rescheduled every job frame sync time. The priority level of the job is determined by the time left until the end of that job frame. Thus, the job with the least-time-to-go will have priority 1, the job with the next least-time-to-go is given priority 2, and so on. If a job attempts to use more CPU time than specified as RCT for that job, the system will abort CPU utilization of that job until the next frame. This ensures system integrity by preventing one job from affecting another. This algorithm can also be easily implemented by hardware or software. Many 6000 time critical simulation systems have been implemented utilizing a dynamic software scheduler.

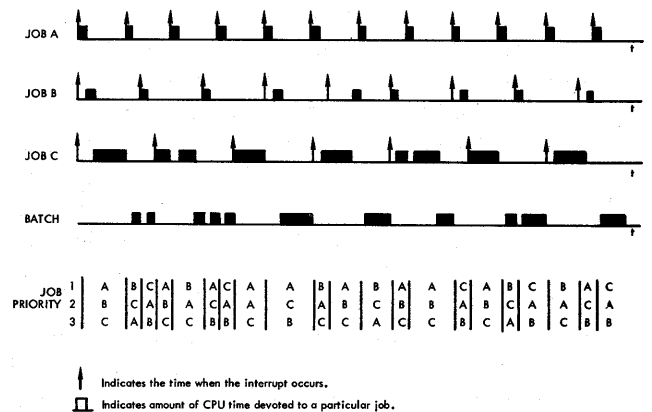


Figure 5—Least time to go—Job basis dynamic scheduling

Least-time-to-go interrupt basis

The basic least-time-to-go interrupt basis (LTTG-IB) scheduling algorithm is identical to LTTG-JB with the exception that every interrupt of each job is scheduled. For example, assume a system running with three jobs, each job using ten interrupts. In this example, the LTTG-JB algorithm would schedule against a single specified FT which is short enough to provide the response required by all ten interrupts in the job. Also, the specified RCT will contain total job RCT requirements for all interrupts in the job. Thus, scheduling is performed against the three groups of job parameters.

When utilizing the basic LTTG-IB algorithm, FT and RCT are specified for each interrupt; therefore, scheduling in this example would be performed against all thirty groups of interrupt parameters.

Up until this point, discussion has been of a basic LTTG-IB algorithm which utilized the FT and RCT parameters specified in the section on LTTG-JB. This algorithm, however, lends itself to a more powerful scheduling mechanism by redefining the two parameters of FT and RCT into three parameters called Tolerance, Period, and Real Compute Time (Figure 6).

- Tolerance (T)—The parameter that “least-time-to-go” is scheduled against. In the synchronous periodic situation, T will usually be equivalent to FT in the basic LTTG-JB algorithm.
- Period (P)—Specifies the minimum period in which this interrupt can occur again. For periodic synchronous interrupts, P will usually equal T.
- Real Compute Time (RCT)—The total CPU time specified to be completed in the T allowed.

These parameters provide efficient scheduling of normal periodic synchronous interrupts, periodic interrupts demanding fast response times, and also allows a means for the scheduling of asynchronous interrupts.

The following example illustrates the scheduling this algorithm provides. Assume a two-job system with two interrupts per job and parameters as defined in Figure 7. Recall that T is the parameter that “least-time-to-go”

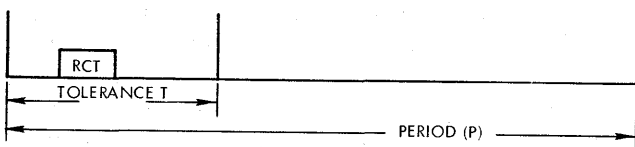


Figure 6—Tolerance parameter

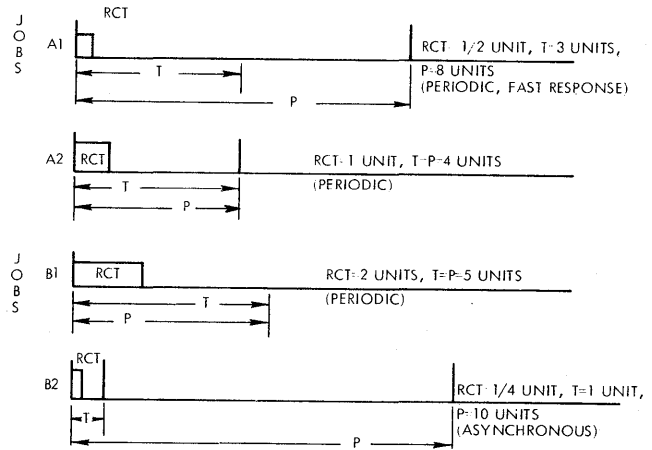


Figure 7—Tolerance, period, and real compute time, job examples

is scheduled against, and P is the parameter which prevents an interrupt from being scheduled more frequently than the user has specified. Figure 8 illustrates a dynamic scheduling of the system. Again, it is to be pointed out that the system will not allow an interrupt to overrun its RCT specifications, and all unused CPU time is available for batch work.

The quantity of parameters (up to 64 in the current implementation) which must be scheduled against one another makes implementation of this algorithm by a software scheduler impractical. The hardware implementation of this scheduler is presented later.

CPU exchange process

Each independent job running in the system requires that a number of items of information be saved and

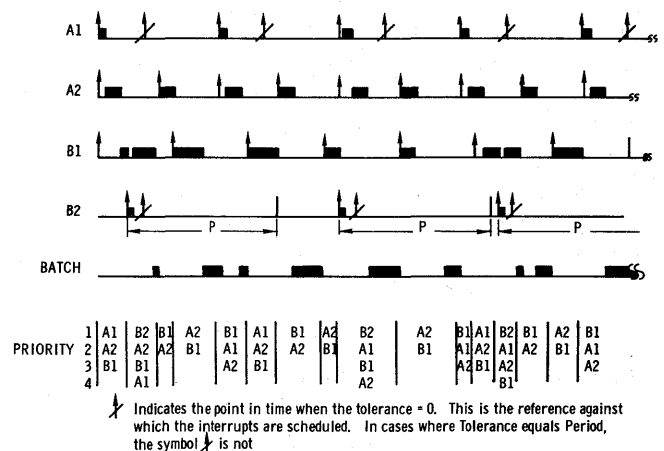


Figure 8—Dynamic scheduling

maintained. Such information includes the contents of all CPU registers, the control card buffer, the buffer containing the job's history of events, messages, charges, the time limit and times consumed, and many other flags relating to the status of the job.

Because information for each job is independent of other jobs, the items are grouped into control points. An active job is always signed on at a unique control point, and all software related requests and operations within the system are linked to the respective control point.

The time critical operating system normally provided runs seven control points (1 through 7) for user jobs and utility operations. (A version of the operating system which provides for 15 control points is available.) The system maintains other control points for internal operations. Each of the seven control points can run an independent job using any desired external equipment without interference, up to the limits of the system resources. Any number of these control points may be used to execute concurrent real-time jobs providing sufficient resources are available.

The Exchange Jump instruction is used by the system for interrupting the CPU. This instruction allows a complete exchange of the CPU executing environment in 5 μ sec. For this reason the Exchange Jump is used for the processing of interrupt routines and for sharing the CPU by control points.

The system reserves space for one Exchange Jump package for each control point. The CPU registers for a control point which is not executing (idle, waiting for I/O, or waiting for a higher priority program to complete) are stored in this exchange package area. Additional Exchange Jump packages may be defined by the system programmer, one for each requested interrupt. When that interrupt occurs, the system initiates the user's interrupt routine by using that interrupt's exchange package.

In the hardware monitor system all incoming interrupts are recognized by the hardware real-time monitor (HRTM). A hardware scheduler in the HRTM schedules the new interrupt against the ones currently in the schedule using the LTTG-IB algorithm. If a CPU task change is indicated, the EXJ controller in the HRTM locks out all batch processing exchange and requests and issues the EXJ instruction through a hardware modification directly to the CPU.

Theory of operation of real-time jobs

A job is submitted with the real-time parameter (RT) on the job card to distinguish the job from normal batch jobs. The real-time job is then held in the input

queue until it is directed to a cleared control point. The job usually consists of three records. The first record contains control cards. For each group of special equipment to be used by the job, a file must be created using the standard REQUEST control card. Other control cards are used to perform tasks associated with the job. The second record contains the program. Files used by the program must be defined in the program. The third record is the data record. It must contain the real-time control cards in addition to any other necessary data. A job enters real-time mode with a FORTRAN call (SIM RUN). Prior to placing the job in real-time mode, the system determines if the program will adversely affect other real-time jobs by performing the static schedule check and a check for availability of interrupt hardware. If other jobs will not be affected, and if the requested hardware is available, the system accepts the job into real-time status, clears all interrupts associated with the job, and control of the execution of the time critical job is passed to the Hardware Real Time Monitor and Central Resident Monitor (CRM).

After acceptance into real-time and upon occurrence of an interrupt, the following sequence of events is initiated:

- The hardware monitor begins decrementing the interrupt's tolerance counter.
- The interrupt is passed on to DADIOS where all channels associated with that interrupt are converted.
- An "End of Convert" signal is passed on to HRTM which causes the start of scheduling.
- HRTM performs the dynamic schedule comparing the remaining tolerance of all tasks waiting for or using the CPU.
- If the remaining tolerance for this task is less than all others, HRTM issues an exchange jump to CRM.
- CRM receives the number of the most critical task from HRTM and, after performing certain accounting functions, places it in execution.

When the task completes execution, it returns to CRM by calling from FORTRAN (SIM WAIT or SIM IDLE), which informs HRTM that processing for this interrupt is complete for this period.

If HRTM detects an RCT overrun (i.e., a task attempting to use more than its assigned CPU time) from real-time status, or if a synchronous interrupt attempts to interrupt too often, an error condition occurs and the action taken is dependent upon the mode selection in the original FORTRAN SIM RUN call.

The user may exit from real-time mode by calling

SIM STOP or by encountering some unrecoverable error condition. After exiting, the user's control point returns to batch mode. He may now do post-processing, or he may return to real-time by calling SIM RUN.

Time critical programming language

The standard CDC FORTRAN Extended Compiler has been modified to distinguish between normal variables and those that are assigned to DADIOS (virtual memory). The compiler distinguishes between the individual DADIOS channels (ADC's, DAC's, etc.) as well as the particular function on each channel (i.e., with or without fix/float conversion). Finally, interrupt definition can be specified. The user is able to address DADIOS channels either directly or indirectly. The linking of indirectly addressed channels is accomplished at run time so that changes in physical channel assignment can be made without requiring recompilation.

Interrupt specification

Interrupts are specified by the statement form:

INTERRUPT

(I = *n*, H = *m*, S = *i*, R = *x*, T = *y*, P = *z*, E = *j*)

where

- n* = logical interrupt number.
- m* = hardware interrupt number.
- i* = interrupt set number.
- x* = required compute time for the interrupt (in units of 10 μ secs).
- y* = tolerance of the interrupt (in units of μ secs).
- z* = period of the interrupt (in units of 10 μ secs).
- j* = external interrupt indicator.

The set designator allows a means to define an interrupt with several "sets" of descriptors (RCT, PER, TOL). At the beginning of a real-time job, all sets of interrupt descriptors are statically scheduled against all other sets of interrupt descriptors in the system. If the schedule is successful, it allows the user to switch from one set of descriptors to another during real-time without having to reschedule.

Loader control card

The entry point for a particular interrupt is specified by the loader control card

INTSEGM(*n*)

where

n is the logical interrupt number.

The loader control card INTSEGM is recognized by the compiler if it appears between subprograms. Compiler processing places it in the desired position on the binary output file. The loader, in turn, associates interrupt number-*n* with the next entry point it encounters, and places this information in a table for use at initialization time. An example of its placement is presented later in this section.

DADIOS variable specifications

Variables to be assigned to DADIOS channels are specified in labeled common blocks. The functions to be performed (fix/float) are based on the type of the variable (integer, real, etc.).

DADIOS variables are handled differently, in that instead of allocating core to these variables, DADIOS channel addresses are assigned to them. When these variables are used (at run time), hardware will route the data over the appropriate channels to/from the DAC/ADC's, etc.

The variables are specified by the statement form:

COMMON/ { *ADC_{*n*}
*DAC_{*n*}
*IDIS_{*n*}
*ODIS_{*n*} } /*k*₁, {*a*₁, ..., {*k*_{*i*}·} *a*_{*i*}

where

- n* is an integer constant specifying a DADIOS unit number ($1 \leq n \leq 4$);
- k*_{*i*} is an (optional) integer constant which specifies the channel (relative to the first channel for this type) to be associated with the following *a*_{*i*} (default = 1);
- a*_{*i*} are variable names, array names, or array declarations in which one, two, or three constant dimensions are specified.

DADIOS variables can be typed either implicitly by the labeled COMMON statement or explicitly by the TYPE statement, as standard FORTRAN conventions allow. The type (integer or real) is reflected in the variables address (1 bit). The hardware then determines whether or not conversion is to be made.

In order to obtain the desired interrupt/element association, the main program must contain specifications for all interrupts to be used. For each interrupt,

the interrupt statement must be followed by the associated DADIOS variable statements.

Subroutines associated with each interrupt are required to have the corresponding DADIOS variable statements, but do not require the interrupt statements.

Example:

```
PROGRAM TEST (INPUT, OUTPUT, HFILE)
```

NOTE 1

```
DIMENSION—
```

```
DATA—
```

```
.
```

```
.
```

```
.
```

```
INTERRUPT (I=1, H=1, R=1000,  
T=2500, P=2500)
```

```
COMMON/*ADC1/A(32)
```

```
COMMON/*DAC/33, B(32)
```

NOTE 2

```
INTERRUPT (I=2, H=3, R=500,  
T=2500, P=3000)
```

```
COMMON/*ADC2/C(32)
```

```
COMMON/*IDIS1/ID(10)
```

NOTE 3

```
.
```

```
.
```

```
EXECUTABLE STATEMENTS
```

```
.
```

```
.
```

```
END
```

```
INTSEGM(1)
```

NOTE 4

```
SUBROUTINE INT1
```

```
COMMON/*ADC1/A(32)
```

```
COMMON/*DAC1/33, B(32)
```

NOTE 5

```
.
```

```
.
```

```
END
```

```
INTSEGM(2)
```

NOTE 6

```
SUBROUTINE INT2
```

```
COMMON/*ADC2/C(32)
```

NOTE 7

```
COMMON/*IDIS1/ID(10)
```

```
.
```

```
.
```

```
END
```

NOTE 8

NOTE 1: Information pertaining to the hybrid environment is maintained in file HFILE for system usage.

NOTE 2: (Logical) interrupt 1 is to be associated with hardware interrupt 1, an internal interrupt (by virtue of default value of omitted E parameter), hence synchronous,

to have an RCT of 10 ms, a TOLERANCE of 25 ms, and a PERIOD of 25 ms. The first 32 ADC elements of DADIOS unit 1, as well as the 32 DAC elements, starting at element 33, of the same DADIOS unit, are to be associated with interrupt 1.

NOTE 3: (Logical) interrupt 2 is to be associated with hardware interrupt 3, also synchronous, and is to have an RCT of 5 ms, tolerance of 25 ms, and a period of 30 ms. The first 32 ADC elements of DADIOS unit 2 and the first 10 input discrete channels (each 16 bits wide) of DADIOS unit 1 are to be associated with this interrupt.

NOTE 4: This loader directive will associate the primary entry point of subroutine INT1 with interrupt 1, and will cause all unsatisfied externals, up to this point, to be satisfied.

NOTE 5: This is a duplication of the element specification to establish addresses for the DADIOS variables A and B for this (interrupt) subroutine.

NOTE 6: This loader directive will associate the primary entry point of subroutine INT2 with interrupt 2, and will cause all unsatisfied externals, since the last INTSEGM directive, to be satisfied.

NOTE 7: These statements establish addresses for DADIOS variables C and ID for this subroutine.

NOTE 8: In addition to standard end-of-program processing, all unsatisfied externals since the last INTSEGM directive will be satisfied.

HIGH PERFORMANCE LINKAGE— HARDWARE

The High Performance Linkage System as shown in Figure 9, uses a DADIOS (Direct Analog Discrete Input Output Subsystem) to gain fast data transfer, an ACS (Analog Control Subsystem) to control any analog devices in the system, a HRTM (Hardware Real Time Monitor) to efficiently schedule "interrupts" into the system together with a CB (Control Board) to tie system control functions together. The tie of the system to the 6000 mainframe for data transfers is via a Data Bus Extension modification to the mainframe and a Bus Adapter which allows up to four 6000 CPU's to access the linkage. The tie of the interrupt scheduling structure of the HRTM to the 6000 mainframe is via an exchange jump modification

to the mainframe. These devices are discussed in the following paragraphs.

Mainframe modifications and bus adapter

The Data Bus Extension modification to the mainframe extends the central memory bus to the outside world and contains the hardware to allow this extension to function as an extension of central memory of the 6000 mainframe. The Bus Adapter adapts this extended memory port (together with identical ports on up to four 6000 mainframes) to the DADIOS, HRTM, and ACS. This device's function is primarily one of timing and resynchronization.

The Exchange Jump modification to the mainframe provides an external port into the exchange jump mechanism of the 6000 mainframe.

Dadios

The DADIOS is a multi-programmed linkage subsystem which allows n jobs ($n \leq 15$) running in the system to concurrently utilize DADIOS for data conversion. Note, that the system may connect to as many as 4 CDC 6000 or CYBER 70 mainframes. DADIOS is a combination of linkage modules providing the following capabilities:

- Simplified programming for data transfer to/from central processor unit and DADIOS on extended central memory read and write buses.
- Access from peripheral processors via standard data channel for setup.
- Individual assignment of channels (in groups of 8) with each job number under program control

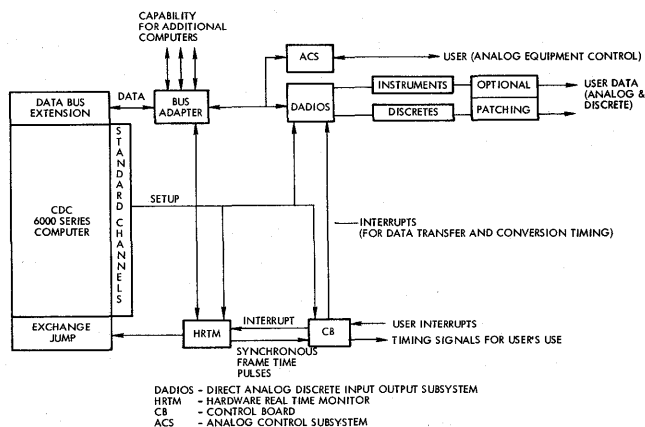


Figure 9—High performance linkage system

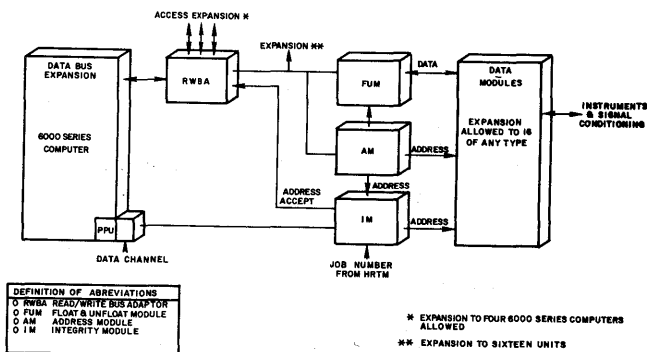


Figure 10—DADIOS

(element reservation of ADC's, DAC's and discretes).

- Data integrity between concurrently processing jobs.
- Program selectable hardware fixed-to-floating point conversion for all ADC channels.
- Program selectable hardware floating-to-fixed point conversion for all DAC channels.
- Intermixing of various types of instruments.
- Expansion capability.

An overall block diagram of DADIOS is shown in Figure 10. DADIOS can be addressed from the CPU and pass data to or from the CPU through the Bus Adapter and Data Bus Extension.

The Bus Adapter and Data Bus Extension are transparent in that they do not change addresses or data in any way. In effect, DADIOS communicates directly with the CPU with its primary purpose being to pass data between the outside world and the CPU. The path of DADIOS via the Data Channel is primarily used for DADIOS setup.

As indicated by Figure 10, DADIOS is comprised of a series of modules which can be configured into a linkage system as required. Brief descriptions of each module indicated in Figure 10 are given in the following paragraphs.

Address module

Any channel in each of the input or output modules can be addressed. The address carries a bit to indicate which fixed-to-floating point or floating point to fixed point conversion is to be done.

Float and unfloat module

Sixteen-bit fixed point data words are converted into 60-bit floating point words (and vice versa) in this module.

Integrity module

Integrity between jobs is provided by this module so that one job cannot interfere with the operation of another job in any way or at any time. For example, the module prevents a user from altering data in any channel not assigned for his use.

Data modules

Four types of Data Modules (Serial Input, Parallel Input, Serial Output, Parallel Output) can be provided. Each module provides buffering for 16 data words each 16-bits wide. Data can be routed to or from these modules through the Float and Unfloat Module or directly from the CPU or instruments.

The data flow through the DADIOS system is illustrated by the following paragraphs.

Data input

Since data is not stored in central memory, but rather in the DADIOS interface, the sequence of events for acquiring the data is straightforward.

1. A sync pulse for the interrupt is issued to the DADIOS system.
2. DADIOS has the capability to utilize one A/D converter per analog channel. In this case, the interrupt would issue a start convert pulse to all channels previously assigned to that interrupt and they would all convert simultaneously. When data conversion is complete, the data can essentially be thought of as residing in central memory.

If a multiplexed analog system is used as instrumentation on DADIOS, the interrupt will place the applicable sample and hold units into hold mode. The multiplexer-converter would then start converting the analog data and placing it into the applicable digital buffer for that channel. When all channels associated with the interrupt have been converted and the data is contained in the digital buffer, the data is directly addressable from the central processor.

3. Memory references by the CPU can now act on the data contained in DADIOS, and the references will indicate whether the data is to be presented in the CPU in fixed or floating point format.

Data output

Again it is important to understand that DADIOS is an extension of central memory. The registers of the D/A converters and the discrete signal buffers serve as CM locations into which data can be written. The sequence for writing the data is as follows:

1. The program executing in the CPU writes data to the first rank register of the addressed D/A or discrete location. (A bit in the address will indicate whether or not the data is to be converted from floating-to-fixed point via the floating-to-fixed hardware.)
2. The sync pulse will transfer data from first to second rank registers. In the case of analog channels, the second rank data is converted and presented as an analog voltage.

Analog control subsystem (ACS)

For systems where control of analog computing devices (i.e., analog computers) is required, the ACS concept has been developed to allow this control to be accomplished by the central processor. As indicated by Figure 9, ACS is also treated as an extension of central memory. Thus, if a mode is to be changed, a pot or digital coefficient unit is to be set, a DVM reading is to be made, etc., these operations are initiated by writing into the ACS section of "extended memory" and any information to be received from the analog device is received via reading from the ACS section of "extended memory."

Hardware real-time monitor

The Hardware Real Time Monitor (HRTM) is designed to provide faster response to external interrupts of real-time jobs than can be done with a software monitor system. This is accomplished by performing the detection and dynamic scheduling between external interrupts in hardware circuits. The following capabilities are provided with the HRTM:

- Scheduling and monitoring of up to 15 concurrently executing real-time hybrid simulation jobs.
- Dynamic scheduling between all interrupts on a "least-time-to-go interrupt basis."
- Assignment of any interrupt to any job, providing more efficient usage of external interrupts.
- Use of three scheduling parameters (real compute time, tolerance, frequency of occurrence) to allow more CPU time for real-time jobs.

- A maximum of 64 external interrupts (a minimum of eight, expandable in groups of eight), terminated on a control board (for user termination).
- Availability of providing data to allow time-accounting on an interrupt basis.
- Access to HRTM via Data Bus Extension (used for central monitor information and error status to central monitor and user).
- Access to peripheral processor via standard data channel for setup and monitor functions.
- Error detection circuitry to prevent one time critical job from interfering with another.

System design

A diagram showing how the HRTM fits into a real-time system is shown in Figure 9. The HRTM can gain access to the CPU by directly providing an exchange jump to the mainframe. The CPU has the ability to read directly from the HRTM through the extended memory via the Read/Write Bus Adaptor. Any PPU may setup the HRTM and input status information via the Data Channel Adapter. The HRTM receives all interrupts and returns all control signals to a control board.

System description

An overall block diagram of the HRTM is shown in Figure 11.

Error detection

The Error Detection circuitry is necessary to ensure that one user does not interfere with another. The enable/disable logic will prevent any unused interrupts from requesting processing. The following parameters

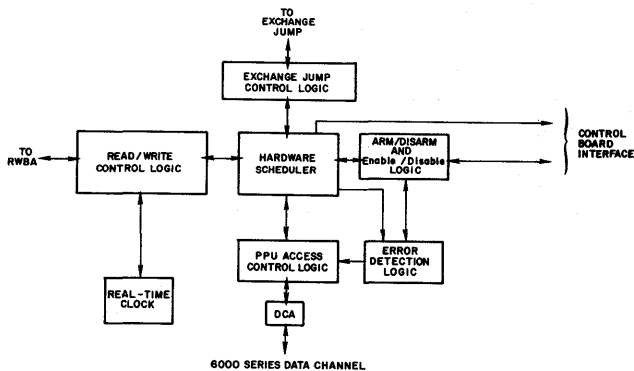


Figure 11—HRTM block diagram

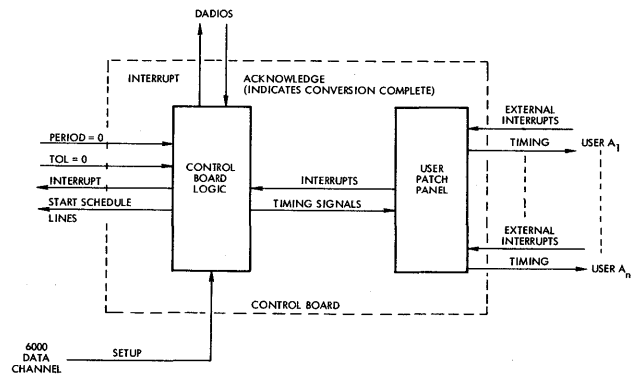


Figure 12—Control board

are monitored and if any errors are found, the job is removed from real-time.

- The required compute time of each interrupt is monitored and countered down. If the RCT for an interrupt counts to zero and the computation is not complete, the job is considered in error and it will not be allowed to continue until its next frame. The user will not be notified of this condition and he can take any action he wants; however, it is noted that not allowing him to continue until the next frame protects other users from feeling any effects of his overrun of his schedule.
- The occurrence period for each interrupt is monitored, and if interrupts occur faster than the specified period, the job is considered in error. The HRTM will not allow interrupts to be scheduled faster than the specified period. User will be notified of the error condition.
- Error signal lines from the DADIOS interface system are sent to the HRTM via the control board. These are errors which occur if the operating program attempts to write out to a DAC or output discrete, or read an ADC or input discrete assigned to another job number. They are brought to HRTM so errors can be associated with the interrupt which had them.

Control board

The Control Board (CB) is the device in the linkage system which allows the system to be tailored to a particular user's needs.

All Period and Tolerance Parameters (used for generation of synchronous time frames) from and interrupt schedule lines to the HRTM, user external interrupt signals, timing signals to user, and interrupt

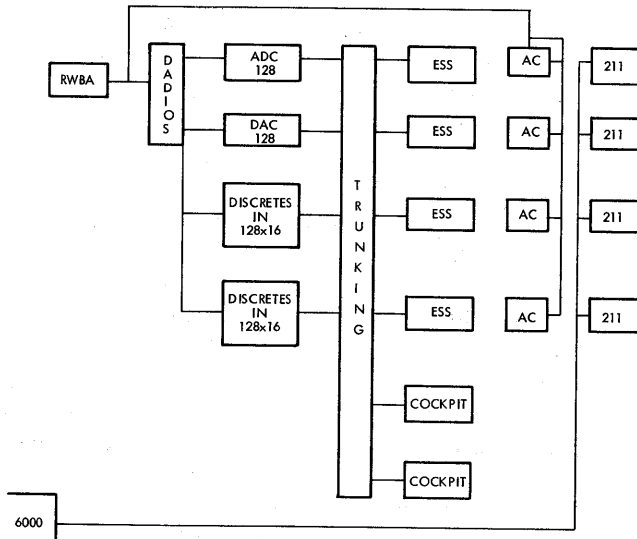
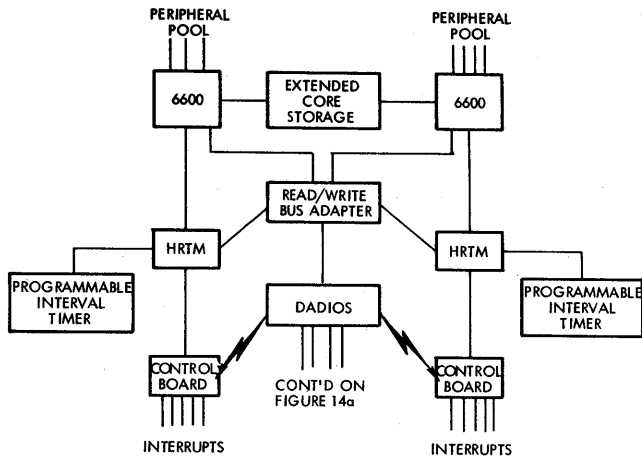


Figure 13—NADC computing system

signals to the DADIOS are terminated on the Control Board. Figure 12 indicates typical ties to the Control Board.

The basic reason for the Control Board is to adapt the flexibility of the system to the user's needs. The more flexible the user wants to be, of course, the more combinations of paths between the user, DADIOS, and HRTM will exist and the more complicated the Control

Board will be. Examples of some of the basic functions that can be provided by the proper use of the Control Board are:

- Ability under program control to assign any interrupt source to a particular portion of DADIOS equipment.
- Ability to assign an interrupt source either to an external source or to an internal source (usually a Tolerance=0 or Period=0 signal from HRTM).
- Ability to hold up the scheduling of an interrupt until the data has been converted. This is accomplished by:

1. The incoming interrupt is sent to DADIOS to initiate any conversion which is to take place.
2. After all conversion is complete, the acknowledge signal is sent from DADIOS to the Control Board and then to HRTM to start the dynamic scheduling of this interrupt. This process permits the conversion of data to completely take place prior to involving the CPU with the interrupt task and prevents the occurrence of a CPU wait for data dead spot.

The NADC computing system

The Naval Air Development Center, Johnsville, Pa., is acquiring a large computing complex to perform time critical simulation studies as well as scientific batch processing. The NADC configuration is shown in Figure 13.

The complex is currently expanded to handle eight interrupts in each HRTM and 128 channels and 128 D/A channels in the dual 6600 complex.

REFERENCES

- 1 M FINEBERG O SERLIN
Multi-programming for hybrid computation
Proceedings 1967 Fall Joint Computer Conference
- 2 *Time critical simulation systems—General information manual*
Control Data Publication No 44629200

Effective corporate networking, organization, and standardization

by PAUL L. PECK

The MITRE Corporation
McLean, Virginia

INTRODUCTION

As investment in automatic data processing systems has increased, methods to improve the productivity of these systems have constantly been sought. One of the most promising methods is networking—the integration of a number of independent data processing installations connected by data communications to provide improved data processing support for the linked installations.

This paper discusses the advantages of networking, addresses the advantages of utilizing homogeneous configurations in establishing a corporate ADP network and presents a management concept and a proposed network standards guide which it is believed will promote the acceptance, growth and effectiveness of the corporate ADP network.

BACKGROUND

Corporate network development has been hindered by compatibility limitations. Traditionally, data processing has been considered a support function, and decisions on the level and type of data processing support were made at the installation level not at the corporate level. In many companies there was a tendency to deplore the proliferation of incompatible computer systems and data banks as new systems were installed, but to do nothing to achieve compatibility since this was not a corporate objective. Compatibility denotes the ease with which a program running on one system can be transferred to another or the ease that data generated in a particular system format can be utilized by another system. A popular conception is that compatible computer systems will accept programs written in standardized languages and perform the same computations producing the same results from the same data. Because of compatibility limitations, the option of quickly and economically creating an efficient

corporate network with its associated advantages of workload sharing, data sharing, program sharing, remote service, program exchange and joint program development was closed to corporate management.

The incompatibility among systems produced by different vendors, as well as incompatibility among successive systems produced by the same vendor, stems from differing approaches to hardware and operating system design and is magnified at each installation by configuration differences, utilization of assembly rather than procedure-oriented languages, and the introduction of installation-peculiar operating procedures.^{1,2,3,4}

Much of the existing hardware incompatibility stems from differences in character/word lengths, internal computer codes (BCD, ASCII, EBCDIC, etc.), boundary alignment considerations (padding, packing, justification, etc.), error checking techniques and numeric representation alternatives (binary, floating point, etc.).

In addition to these hardware differences, basic software differences exist in the areas of operating systems and languages. Each manufacturer provides a specific operating system to use with his hardware. These operating systems offer widely differing services in procedure-oriented language and utility support, hardware support, file management and input/output control, systems services and job control. Assembly languages differ extensively with regard to the size and nature of the instruction sets and optional features provided.

Data compatibility as such rarely exists. Existing corporate data banks are often not compatible because each installation data base was developed using unique hardware and software. The basic data definitions, data formats, and data structures were designed to satisfy local, not corporate requirements. This has necessitated the development of either special data bases or special programs for format translation to satisfy reporting and interface requirements. Standard means of de-

scribing data elements and standard approaches to data bank development have not been used in the past.

To highlight the extent of the incompatibility that presently exists (even among systems designed to support a common objective), consider the following facts determined in the analysis required to support the World Wide Military Command and Control System (WWMCCS) procurement which will upgrade the processing capabilities of up to 109 existing ADP centers. According to Phil Hirsch,⁵ the ADP centers to be included in the WWMCCS procurement "are supported by 30 different programming languages (dialects are ignored), and 802 separate programs. At least 75 percent of these are in machine-dependent code, primarily Autocoder. There are 20 FORTRAN programs, 8 in COBOL, and 272 in JOVIAL, which may or may not be standardized dialects. . . . Today each WWMCCS installation is largely self-contained. The workload is handled on a batch basis, using local data bases. Although installations are interconnected, the terminals usually are off-line devices."

Since the WWMCCS systems to be replaced range in age from one to 10 years, this situation is probably similar in type, if not in scale, to that found today in many large, decentralized corporations.

FACTORS THAT EASE THE IMPLEMENTATION OF NETWORKING

Although procedure-oriented languages (POLs) such as FORTRAN (the de facto scientific programming language) and COBOL (the standard commercial programming language) were developed to facilitate programming and to ease system-conversion problems, programmers can now code in languages that are somewhat independent of hardware. Therefore, the use of these languages eliminates many compatibility limitations. Program transferability was further promoted when ANSI sanctioned standard specifications for FORTRAN and COBOL.

With the advent of third generation equipment, intra-system compatibility (compatibility among systems produced by the same manufacturer) became realizable. Compatibility is a major design objective of and is widely promoted by all major computer manufacturers. For example, in 1964 IBM announced the System/360, its new line of compatible computers. Since the System/360 was designed for both upward and downward compatibility, a significant step toward intra-system compatibility had been taken. IBM's new series, the System/370, offers intra-system compatibility and is compatible with the System/360 series. Control Data Corporation explicitly states that programs designed for the CDC 6000 Series will operate

on any CDC 6000 Series configuration. Furthermore, Control Data has announced that its new system, the CDC 7600, will be compatible with the 6000 Series. RCA stresses that its Spectra 70 Series, in addition to being upward and downward compatible, is compatible with the IBM 360 Series.

System/360 plug-to-plug compatible magnetic disk drives, magnetic tape drives, and large core storage units are now available from independent peripheral manufacturers. Plug-to-plug compatibility means that the new device is physically and electrically interchangeable with the IBM peripheral, that neither the peripheral nor the equipment to which it connects requires any modification to effect the replacement, and that no modifications are required to the operating system and user programs.⁶ Thus, a network consisting of IBM systems can maintain its compatible status and still take advantage of peripheral development which provides price-performance benefits.

Perhaps the most significant developments have taken place in the area of data communications. Data transmission is growing at a tremendous rate with a corresponding decrease in its costs as the Bell System improves its digital transmission capability. Private microwave links for data communications are being developed by Microwave Communications, Inc. and Datran. Digital multiplexer and modem advances which have made possible more efficient utilization of existing carrier channels for data communications were spurred by the FCC decision in the Carterfone case which permitted the attachment of customer-provided data sets.^{7,8,9,10}

Significant research has been conducted in the areas of routing, buffering, synchronization, error control, reliability and computer-communications interface. The ARPA network has developed a separate communications processor, the Interface Message Processor (IMP) to connect host computers to the telephone network. The Interface Message Processor is an augmented, ruggedized version of the Honeywell DDP-516, and includes 12K 16-bit words of core memory, 16 multiplexed channels, 16 levels of priority interrupt and logic supporting host computers and high speed modems. Since the ARPA network is a heterogeneous network (a network of dissimilar systems), special hardware interfaces have been developed to connect the IMPs to a wide variety of different hosts.¹¹ The MERIT network is engaged in similar research and has developed a similar communications processor.¹²

NETWORKING

Networking is the integration of a number of independent data processing installations connected by

data communications to provide improved data processing support for the linked installations. Existing networks include the TUCC Network, the Control Data Cybernet Network, the Octopus Network, the TSS Network, the ARPA Network and the MERIT Network.¹¹⁻¹⁸

Network advantages

Potential benefits of networking include improved operational efficiency, increased availability of resources, improved ADP backup capability, and possible reductions in ADP support costs. Since alternate data processing resources are available, a user may realize an improvement in turnaround time by submitting his job to another network node when the local facility is saturated. Integration of activities is inherent in networking; consequently, the growth of common, compatible programs, data bases, and data formats will be promoted and duplication of effort will be reduced.

Corporate data processing costs should decrease because of higher system productivity resulting from the increase in workload sharing, program sharing, data sharing, joint program development and program exchange between installations. Workload sharing, the transmission (either manually or automatically) of a discrete job entity to another ADP installation for execution, tends to eliminate the extremes of underutilization and overloading of individual installations. Program sharing and data sharing are variations of workload sharing. Data sharing permits a user to send his programs to another installation for the purpose of utilizing data there. Program sharing enables a user to take advantage of programs at other installations by sending his data to the program. In all types of sharing the output is returned to the user at his location. Both data sharing and program sharing minimize use of communications facilities. Communications traffic is further reduced by remote service which enables the user to utilize both a program and data at another installation and receive the output at his location. Program exchange is the exchange of techniques, sub-routines or complete programs which can be used without incurring the expense of additional modification.

Since successful implementation of networking has led to improvements in processing capability and to cost reductions,^{13,14,15} corporate officers responsible for data processing should seriously consider the implementation of corporate ADP networks.

No quantitative studies of the advantages of networking are cited because I have not been able to find any. The difficulty in quantifying the utility of networking arises because of the general unavailability of

data. Even when data is available, ongoing network development makes it difficult to determine how much of the improvement in effectiveness is due to implementation of the network and how much is the result of tuning the system. However, as an indication of the utility of networking the reader should consider the experience of the TUCC Network.

The Triangle Universities' Computation Center (TUCC) was established in 1965 as a cooperative venture among three major North Carolina universities: Duke University, North Carolina State University (NCSU), and the University of North Carolina (UNC).

The TUCC network is a homogeneous network (similar systems are linked), the center of which is a System 360/75 with one million bytes of high speed core and two million bytes of Large Capacity Storage, operating under OS/MVT. There are approximately 100 terminals (high, medium, and low speed) in the network. The high speed terminals are a 360/50 and an 1130 at UNC, a 360/40 at NCSU and a 360/40 at Duke. The 360 systems are multi-programmed with a partition for local batch work and a telecommunications partition for TUCC remote I/O services. The medium speed terminals are IBM 2780s (or equivalents) and 1130s, and the low-speed terminals are teletypes, IBM 2741s (or equivalents) and IBM 1050s.

According to M. S. Davis, former director of TUCC, the primary incentive for establishing the TUCC Network was economic. The network was formed because it was believed that a larger ADP system serving the three major universities would provide economy of scale and reduce the effect of the shortage of competent systems programmers.¹⁵ The question is often asked if the three universities are better off with the network than if each university had upgraded its individual ADP system. TUCC officials have estimated that if the net hardware cost of the Model 75 were divided three ways, each member would have an additional \$10,800 per month with which to upgrade his existing system. Each university could then install a Model 50 with 256K memory and a 2314 disk file. The network, however, is realizing substantially more power than would be available with the three separate systems. The throughput of the Model 75 alone is about six times that of a Model 50.¹³

Advantages of homogeneous networks

Networking is not a panacea; however, if several decentralized data processing facilities require upgrading in the same time period, the acquisition of common hardware and software is recommended so that corporate management will preserve the capability

of combining these facilities into a network with minimum expenditure of resources and time.

It is recognized that significant effort is being directed toward establishing heterogeneous networks, e.g., the ARPA Network and the MERIT Network. However, these networks are research-oriented, not profit-oriented. In each of these networks, special communications processors, network control systems and communications-computer interfaces have been developed. In both networks extensive effort has been directed toward establishing a network protocol and the MERIT network has proposed the development of a standard data description language to facilitate transmission of data between computers, systems and programs and to provide a convenient and complete format for the storage of data and its associated descriptor information. Development and implementation of a standard data description language would significantly reduce compatibility limitations, however, information interchange between dissimilar systems presents many problems^{19,20,21,22} and some lead time must be anticipated before such a capability will be implemented.

Workload sharing, program sharing, data sharing, program exchange and joint program development are easier to implement in a homogeneous network because program modification and data translation hardware and software can be kept to a minimum, the cost of developing interface hardware and software can be minimized, and network protocol is easier to implement.

An often-voiced disadvantage of a homogeneous network is that the user is not provided the opportunity to utilize special hardware and software capabilities provided by other vendors. Presently, to take advantage of the special capabilities of a dissimilar system the user tailors his program to comply with the hardware and software requirements of the dissimilar system. The user must familiarize himself with the hardware, software, and operating idiosyncrasies of the dissimilar system to effectively utilize its capabilities. Consequently, the need for these special capabilities must be scrutinized by corporate officials and alternative means of providing these capabilities considered. For example, in the long run it may be more economical to lease time on the dissimilar system or to use a standard system even though it may not be best suited for processing certain types of programs.

In summary, homogeneous networks are best able to satisfy corporate networking requirements because a minimum expenditure of resources and time is required to implement the network.

ORGANIZING FOR EFFECTIVE CORPORATE NETWORKING

Although common hardware and operating systems provide a foundation for the quick and economical development of networks, corporations must establish a corporate ADP focal point which will be responsible for the development and maintenance of a network standards guide. The need for a corporate ADP focal point and the utility of a network standards guide in implementing a network will be evident to anyone who has attempted to run programs at one installation that were developed elsewhere. Since the individual computer installations in a network are frequently staffed by professionals of dissimilar backgrounds and since the goals of the member ADP installations tend to be parochial rather than corporate, some means of facilitating communication and cooperation among the installations is required. A network standards guide defining corporate standards serves as a common reference point for all installations.

The formation of corporate ADP networks has been hindered by the absence of a corporate focal point for standardization. As evidence of this situation, consider that in general:

- short-term individualized solutions, rather than common applications programs have been developed;
- standard programming approaches to specific classes of applications have not been developed;
- standardized benchmarks for the evaluation of programming approaches do not exist;
- ANSI standard procedure-oriented languages are not utilized;
- a variety of documentation techniques exists; and
- compatible data banks are rare.

The creation of a Corporate ADP Coordinating Office and a Corporation User Group will ease the solution of these problems and facilitate integration of individual ADP installations into a corporate network.

Corporate ADP coordinating office

This office, which reports directly to the corporate data processing director is responsible for developing corporate ADP policy and for providing direction and assistance in the establishment and maintenance of a corporate ADP network. Since the corporate data processing director is responsible for controlling and

- Develop the network standards guide.
- Develop benchmarks for the comparison of various approaches to the solution of vital corporate problems and evaluate these approaches.
- Develop and maintain a library of corporate program documentation.
- Provide a vehicle for the dissemination of information among the decentralized facilities.
- Review corporate ADP system needs.
- Monitor and provide assistance in corporate ADP procurement and efforts.

Figure 1—Functions of the corporate ADP coordinating office

coordinating all data processing activities, the Corporate Data Processing Office personnel will serve as functional staff with implied line management power because of the authority of the corporate data processing director. Suggested functions of this office are listed in Figure 1.

No organization is recommended because specific organizational relationships will be developed in accordance with the management concepts of the corporate data processing director. Two networks with centralized management are the Octopus Network and the Cybernet Network. The Computation Department, Lawrence Radiation Laboratory, University of California/Livermore has overall responsibility for operation of the Octopus network and maintains separate project groups for software design and development, program evaluation, documentation dissemination and standards development and maintenance. All activities in Control Data Corporation's Cybernet Network including hardware/software development, resource accountability, and documentation development and dissemination are controlled by the Data Services Division.

A major function of the Corporate Data Processing Office is the development and enforcement of the detailed standards that comprise the network standards guide. This office is responsible for:

- determination of the type and degree of standardization required;
- implementation of the standardization program; and
- management of the program.

To ensure successful implementation of the network, the network standardization program must have the complete backing of the corporate data processing director, the importance that he attaches to the network standards guide must be well publicized and the most effective means of initiating the network stan-

dardization program must be determined. The two basic methods of initiating the network standardization program: the phased implementation approach and the one-step implementation approach must be evaluated. The advantages of each approach (decreased costs, increased effectiveness, etc.), the probability and cost of implementing each approach, and the impact on networking and current installation activities must be determined. If the phased implementation approach proves to be the better choice, the areas to be standardized must be specified and a phasing schedule developed.

The network standards section of the Corporate ADP Coordinating Office will be responsible for the implementation and management of the standardization program. This continuing effort includes the review, modification, and enforcement of existing standards and initiation of new standards as they are needed. Furthermore, to combat organizational parochialism and a breakdown of the network due to blurring of responsibilities, this office must continually review the relationship of individual computer systems to the network to ensure that the overall interests of the company are being maintained. The Corporate ADP Coordinating Office thus serves as the network control mechanism.

Corporation user group

To ensure the effectiveness of the network standardization program, a users' group is needed. This users' group, composed of key installation representatives, will advise and assist the Corporate ADP Coordinating Office in integrating the decentralized ADP facilities into a corporate network. As such, one of their prime functions is to help develop the network standards guide. This participation should encourage coordination and communication among the decentralized installations, thus making it easier to establish the network standardization program. Once the network is established, this group will continue to support the Corporate ADP Coordinating Office by proposing and reviewing network standards as required.

NETWORK STANDARDS GUIDE

Interchangeability of programs and data is fundamental to economical networking and interchangeability is a function of standardization.

Standardization is the process of developing, reviewing, promulgating, and enforcing guidelines for controlling the performance of the discrete elements

Operating Standards

- Network Protocol
- Configuration
- Job Preparation
- Job Processing
- Job Termination

Software Standards

- System Analysis Standards
- Programming Standards

Management Standards

- Data Conventions
- Program Classification Rules
- Networking Rules
- Utilization, Review, and Modification Rules

Figure 2—Network standards guide outline

which interact in the operation of a system. Within an installation, ADP standards facilitate integration of system elements (hardware, software, procedures), make possible reductions in both operating and software costs, ease long range planning, and are a vital element in increasing flexibility. Standards increase the ability of an ADP installation to respond to changing operational needs.

Similarly, just as installation standards are needed to integrate the discrete system elements, network standards are required if installations are to be integrated into a network.

The network standards guide will consist of the following three categories of standards: operating, software, and management. Figure 2 is a proposed outline for the network standards guide and indicates how these categories might be further subdivided.

Operating standards

Operating standards include network protocol, configuration rules, job preparation, job processing, and job termination procedures. Network protocol is the operating rules and procedures to be utilized in the receipt, processing, and transmission of programs and data initiated at other installations. Included here are error control procedures, message transmission techniques, and methods of determining and specifying program priorities.

Configuration rules are needed to ensure that a nucleus of hardware and software compatibility is maintained among the network installations. Although each decentralized installation is required to maintain this degree of compatibility, it is recognized that certain installations may require special purpose hardware or software (e.g., large core storage, graphics systems and

text processing systems) to satisfy local needs. The addition of these special capabilities is encouraged, however, the Corporate ADP Coordinating Office should determine if these capabilities can be effectively utilized by other installations and develop standards for their use if it proves necessary. The maintenance of software compatibility requires continuous monitoring of vendor modifications to the operating system and the support software. Because of the number and variety of special features and peripherals available, a strictly enforced configuration control policy is needed if system compatibility is to be maintained. For example, to maintain hardware compatibility the required computer configuration must be defined and channel assignments and device addresses must be standardized.

Job preparation standards include procedures for preparing both the input data and the programs required for a data processing run. Programs should be categorized and common job submission and job control statements developed. Inherent in the development of these common control statements is the assumption that common default options will be specified for each run category. At program compilation time, a programmer usually has several options which affect ancillary aspects of the compilation (e.g., optimize or do not optimize the resulting machine language program; list or do not list the assembly language code). If the programmer does not specify any options, standard default options are invoked.

Job processing refers to all job functions performed by the computer operator from initiation through termination of a programming run. Initiation involves the preliminary set-up of all peripheral devices, the initial setting of console switches, and the actual run initiation procedure. Execution includes any additional set-up activity that can be overlapped, listing of unusual events and operator messages, and operator responses to interrupts and error conditions.

Job termination standards refer to take-down procedures utilized under both ordinary and abnormal conditions, control of installation data, and disposition of computer results.

Software standards

Software standards refer to those practices which systems analysts and programmers use in their daily work. Adherence to these standards promotes resource sharing and facilitates communication between the systems design, programming, and user functions. Systems analysis standards are those practices that ease the preparation of the system specification. The level of

detail of a good system specification is such that the structure, functions, flow, and control of the system is defined so that a programmer can readily program, test, and implement the system.

Programming standards are those guidelines used by programmers in their daily work. As data processing capability has increased, the methods of best utilizing this capability have changed. To increase the overall effectiveness of ADP support, good programming standards are necessary. Modular program design techniques (so that the program can be handled as a series of subtasks), standard programming approaches to the solution of well defined applications, and common testing and checkout procedures should be utilized. Frequently, a number of software programs are available for solving certain types of problems (e.g., square root and trigonometric conversion routines). The development of benchmark tests to evaluate these approaches will promote efficiency since the advantages and disadvantages of each technique will be determined and a network standard will be developed.

It is generally agreed that there is a need to provide information on what a program is supposed to do and how it does it. Recognizing this, ADP installations have independently developed a variety of program documentation guidelines, resulting in a lack of real standardization in documentation. Programming documentation consists of recording the detailed logic and coding of a program. Entire books have been written on programming documentation,^{23,24} and documentation standards have been established by ANSI. At a minimum, programming documentation should consist of a summary change log, general program description, logical and mathematical description, detailed program flow charts, detailed program, file, and data descriptions, an assembly listing, a run book (providing calling sequences and job control and configuration requirements, etc.).

Management standards

Before the operating and software standards discussed above can achieve maximum effect, a framework must be developed for their utilization. This framework will consist of general policies to promote conformity of usage and will ultimately determine the effectiveness of networking for workload sharing, program sharing, data sharing, remote service, program exchange, and joint program development.

Management standards can be categorized as data conventions, programming classification rules, networking rules, and utilization, review and modification rules.

Data conventions

Data conventions ease data exchange because they insure that the data itself is consistent.

Standardization guidelines are needed in the areas of:

- data element definition;
- terminology;
- definitions of values and constants;
- data names;
- data exchange formats; and
- data file structures.

The ASCII (American Standard Code for Information Interchange) code should be used for all data to be transferred between facilities in order to reduce the number of code translation programs each facility must maintain and to ease the installation of non-compatible terminals if the need should arise in the future. Common data exchange formats will further reduce the complexity of the translation programs.

Program classification rules

One of the benefits of networking is that common program development and program exchange will be encouraged. If program exchange is to be effective, the amount of effort required to search for available programs, subroutines, and approaches must be reduced. Each ADP installation must therefore categorize its programs. After this has been done, either standard categories, with clear definitions of what is included in each classification, could be developed and all programs classified; or each facility could distribute copies of its categories, category definitions, and the programs which fall into each category. Program classification (a first step in eliminating duplication of effort) is essential if common program development and program exchange are to yield maximum gains and if software conversion costs are to be kept to a minimum.

Networking rules

Guidelines must be developed which describe the administrative practices and technical restrictions required in a network environment. These include priority determination rules, restrictions on the mixing of languages within a program, incentives to use ANSI FORTRAN and COBOL, and configuration rules which restrict programmers to utilization of a subset of the available facilities (core, peripherals, etc.).

Utilization, review and modification rules

These management standards define the conditions under which the various standards apply, the review cycle and the steps in the standards modification process. Utilization rules must consider the needs of the network yet make allowances for the differences which exist between installations. For example, installation-peculiar programs need not be subject to the strict rules which apply to programs which will either be run at or utilized by other ADP installations. The standards section of the Corporate ADP Coordinating Office together with the Corporation User Group, should develop and implement formal review and modification procedures.

IMPLEMENTATION OF THE NETWORK STANDARDS GUIDE

The implementation of a standardization program for a single installation requires a significant amount of preparation and a great deal of marketing ability. Additional difficulties are encountered when standards are to be developed that apply to more than one ADP facility. These difficulties are magnified when the separate facilities are to be integrated into a network. In order to facilitate the introduction of the network standards guide, the following plan is suggested:

- (1) Form a standardization team consisting of competent representatives from each installation which is to be assimilated into the network.
- (2) Announce the standardization program with a letter from corporate headquarters and a personal visit by the corporate data processing director to each decentralized ADP installation.
- (3) Develop an outline for the network standards guide (e.g., an expansion of Figure 2).
- (4) Submit the outline to the ADP installation managers.
- (5) Develop the actual standards:
 - survey existing standards;
 - determine which standards should be kept and improved;
 - develop the new standards which are needed;
- (6) Submit these standards for review and approval by the ADP installation managers.
- (7) Distribute copies of the standards to all staff members of all ADP facilities.
- (8) Arrange open meetings at each installation so that the corporate data processing director and the members of the standards section and

Corporation User Group can explain how the standards program was developed.

- (9) Initiate training for each ADP installation staff group (programmers, operators, etc.) which will be affected by the standards.
- (10) Implement the standards.
- (11) Establish a formal network standards review committee composed of corporate and installation data processing personnel.
- (12) Enforce the standards:
 - incentives to abide by the standards should be given to each ADP facility by the Corporate ADP Coordinating Office;
 - incentives should also be given to staff members of each ADP facility by their managers; and
 - continual monitoring and guidance should be provided by both the Corporate ADP Coordinating Office and the managers of the decentralized installations.

SUMMARY

The benefits of workload sharing, program sharing, data sharing, remote service, program exchange, and joint program development are such that networking should be seriously considered by corporations with decentralized ADP facilities. Effective networking provides improved capability, improved operational efficiency and possible reductions in ADP support costs.

Corporate network development has been limited in the past by compatibility limitations, however, the effect of these limitations has been diminished by the development of families of compatible computer systems. Homogeneous networks are proposed as the best method of satisfying corporate networking requirements because a minimum expenditure of resources and time is required to implement the network.

Effective networking requires extensive standardization. It is recommended that a Corporate ADP Coordinating Office and a Corporate User Group be formed. The establishment of a Corporate ADP Coordinating Office will provide a focal point for implementation and management of the network and the creation of a Corporation User Group is fundamental to the creation and continued updating of a relevant network standards guide.

The network standards guide may be divided into operating standards, software standards, and management standards. The latter part of this paper details the functions to be standardized and suggests a plan for implementation of the network standards guide.

ACKNOWLEDGMENTS

The author is grateful to J. J. Powell, P. H. Messing, J. J. Peterson and S. A. Veit for their many suggestions and thoughtful review of this paper.

REFERENCES

- 1 J GOSDEN et al
Achieving inter-ADP center compatibility
The MITRE Corporation MTP-312 May 1968
- 2 J A WARD
A panel session—software transferability
Proceedings of the AFIPS Spring Joint Computer Conference Vol 34 pp 605–612 1969
- 3 K SATTLEY R MILLSTEIN S MARSHALL
On program transferability
RADC Technical Report TR-70-217 November 1970
- 4 P L PECK
The implications of ADP networking standards for operations research
Proceedings of the U.S. Army Operations Research Symposium pp 269–281 1969
- 5 P HIRSCH
WIMMIX: It's the biggest, but will it be the best
Datamation pp 84–90 October 1969
- 6 C R FROST
IBM plug-to-plug peripheral devices
Datamation pp 24–34 October 15 1970
- 7 R A OHARE
Modems and multiplexers
Modern Data pp 58–79 December 1970
- 8 J E BUCKLEY
A survey of communication tariff developments
Datamation pp 127–132 December 1969
- 9 A R WORLEY
Practical aspects of data communications
Datamation pp 60–66 October 1969
- 10 S J KAPLAN
The advancing communication technology and computer communications systems
Proceedings of the AFIPS Spring Joint Computer Conference Vol 32 pp 119–133 1968
- 11 F E HEART et al
The interface message processor for the ARPA computer network
Proceedings of the AFIPS Spring Joint Computer Conference Vol 36 pp 551–567 1970
- 12 E M AUPPERLEE
MERIT computer network hardware
Courant Institute of Mathematical Sciences Computer Network Seminar November 30 1970
- 13 F P BROOKS JR J K FERRELL T M GALLIE
Organizational financial and political aspects of a three-university computing center
Proceedings 1968 IFIP Congress Vol 2 pp 923–927 1968
- 14 D N FREEMAN J R RAGLAND
The response-efficiency tradeoff in a multiple-university system
Datamation pp 112–116 March 1970
- 15 M S DAVIS
Economics—point of view of designer and operator
Proceedings of the Interdisciplinary Conference on Multiple Access Computer Networks pp 4.11–4.17 April 1970
- 16 W J LUTHER
Introduction to cybernet
Courant Institute of Mathematical Sciences Computer Network Seminar November 30 1970
- 17 J G FLETCHER
Livermore time sharing system—part 1: octopus
Computation Department Lawrence Radiation Laboratory University of California/Livermore December 1970
- 18 B HERZOG
MERIT proposal summary
MERIT Computer Network February 1970
- 19 H S MELTZER H F ICKES
Information interchange between dissimilar systems
Modern Data pp 56–67 April 1971
- 20 C S CARR S D CROCKER V G CERF
Host-host communication protocol in the ARPA network
Proceedings of the AFIPS Spring Joint Computer Conference Vol 36 pp 589–597 1970
- 21 A K BHUSHAN R H SHOTY
Procedures and standards for inter-computer communications
Proceedings of the AFIPS Spring Joint Computer Conference Vol 32 pp 95–104 1968
- 22 J L LITTLE C N MOOERS
Standards for user procedures and data formats in automated information systems and networks
Proceedings of the AFIPS Spring Joint Computer Conference Vol 32 pp 89–94 1968
- 23 M GRAY K R LONDON
Documentation standards
Brandon/Systems Press 1969
- 24 D WALSH
A guide for software documentation
Inter-Act Publications 1969

Multi-dimensional security program for a generalized information retrieval system

by JOHN M. CARROLL, ROBERT MARTIN, LORINE McHARDY, and HANS MORAVEC

University of Western Ontario
London, Ontario, Canada

INTRODUCTION

GIRS is a generalized information retrieval system which permits the creation and modification of a data base, as well as the retrieval of specified data from the base.

This system is data independent and flexible so that the user can fit it to his particular application. Its structure is as equally applicable to the storage of abstracts of technical reports as it is to personnel files. A multilevel protection scheme guarantees security of information against unauthorized examination or modification.

The basic model for data storage is a single-page typewritten record, which can be entered into the file with minimal restrictions placed on its structure. This format has been found useful for storing personnel records, inventory records, quality-control records, and bibliographic entries for information storage and retrieval systems.

The multi-dimensionality of the system's security provisions arises from the fact that password or passwords assigned to users determine

- (1) which subset of ten available processing functions they can exercise (level 1 protection),
- (2) on which portions of records (items) they can exercise these functions (level 2 protection), and
- (3) which records they are privileged to work with, or conversely, which records are locked against them (level 3 protection).

Thus data protection is provided at two levels (2 and 3). Associated with each item name is a protection code which applies to the particular item in all records. A particular item, say, Salary in a personnel file, may be protected so that only certain persons may examine

that item in any record. This is level 2 protection. In addition, each item in each record has a protection code applying only to that one item. Thus, while a person may be authorized to examine the Salary item in general, he may be prevented from examining the salaries of his superiors by virtue of the protection on these items within specific record's. This is level 3 protection.

The system is programmed in Fortran for maximal portability among computer systems, although it is presumed that routines will be written in an appropriate assembly language in the interest of system efficiency and flexibility at a particular installation.

DATA STRUCTURE

Any number of data bases can exist simultaneously, each identified by a unique file name. As shown in Figure 1, a file consists of a series of records, each of which consists of a set of items. The items are further subdivided into elements, the smallest unit of information in the structure.

A file can contain up to 99,998 records, each of which is identified by a 5-digit record number. Each record is further divided into items, with the restriction that all records in a file have the same number of items and that they be present in the same order in all records.

Each item has a unique name within the file, and consists of an integral number of lines (1 to 10) of 72 characters each. Further, the total number of items in a given file must not exceed 10, and the total number of lines in a record must be less than or equal to 20. Each item can be subdivided into elements by use of delimiters.

The items are numbered from 0 to 9, and the lines within an item are also numbered from 0 to 9. Each line in the record is therefore identified by a 7-digit

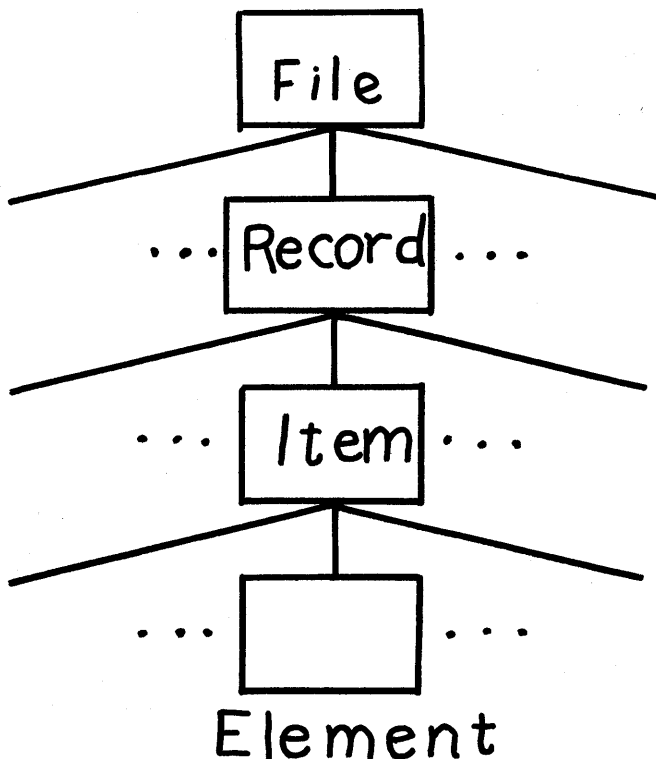


Figure 1—Structure of the data base

number, consisting of the 5-digit record number, a 1-digit item number, and a 1-digit line number within the item.

These restrictions were imposed after observing that the quantitative data in many files can be summarized on a single typewritten face sheet. More narrative data can be held on tape and retrieved in batch for transmission in hardcopy form. Note that an *item* can include several related data elements; elements are coalesced into items on the basis of security considerations.

In a specific implementation record size could easily be made larger if desired.

Two sample data structures are shown in Figure 2. The first is an example of the use of this structure to store abstracts of technical papers. The record consists of 4 items, "TITLE," "AUTHOR," "PUBLISHER," and "ABSTRACT," comprising 1, 2, 1, and 10 lines respectively. The first of these is not subdivided into elements, while the last three are. The second example shows a personnel record format using 6 items and a total of 12 lines. The data used to create a new data base must be contained in a disk file in card image format organized as shown in Figure 3.

The data file is to be written as fixed length Fortran

records of 80 characters in ASCII mode. This file can be read using the random access mode of PDP-10 Fortran. To avoid confusion, the term "record" will refer to one of these 80 character records. The term "data record" will refer to one of the user's input data records identified by a five-digit number. The file is divided into two parts—a 70 record header, followed by the user's data. A diagram of the structure is given in Figure 4.

The header is fixed length and contains all the information necessary to access the data. Its content areas follow:

Records 1-10; passwords and protection keys, 5 sets/record

```

TITLE:    ....title of paper.....
AUTHOR:   last name initials corporate author
          city now with
PUBLISHER: name city pub date Lib. Congress #
ABSTRACT: keywords abstract text.....
          .....
          .....
          .....
          .....
          .....
          .....
          .....
          .....
          .....
          .....
          (A)
NAME:     last first middle position
ADDRESS:  street city
NUMBER:   Soc. Ins. # OHSC # employee # union #
VITA:     date of birth place of birth next of kin
          relationship address
MEDICAL:  physical handicaps,
          allergies,
          etc.
HISTORY:  job record.....
          .....
          .....
          .....
          (B)
  
```

Figure 2—Sample data structures: (a) format of a record for storing bibliographic entries; (b) format of a personal record

Records 11-50; 4 lines per item containing the item name, # lines in the item, level 2 protection code, and up to 24 element names.

Records 51-70; index to data records containing (data record #, Fortran record #) 8 pairs/record.

The index is set up to reference every 160ⁿth record where *n* is the value given on the record count card.

ACCESS PROCEDURE

To access the system, it is necessary to log into the PDP-10/50 host system. This requires use of a project-programmer pair and password. The user then copies to the disk the Generalized Information Retrieval System

password count card

[password detail cards

item count card

[item detail cards

record count card

[record detail cards

termination card

Figure 3—Card input for creation of a file

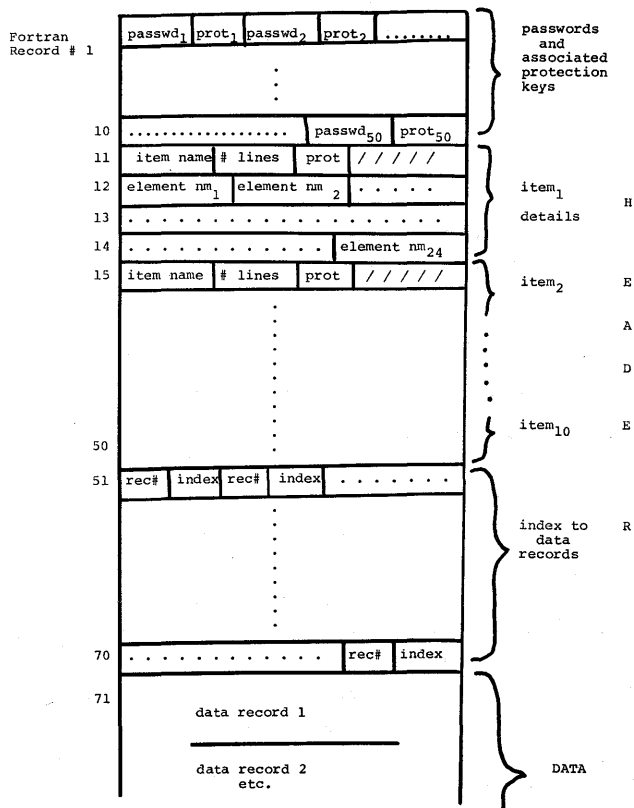


Figure 4—Format of a data file

file and the particular data base file he wishes to use (unless the user is going to create a new file).

If the desired file is found (or a new one created), GIRS will request:

“PASSWORD?”

The user must then type in his password for the GIRS system. Conventional print inhibit provisions for password protection are provided by the host system.

The password is matched against a list made up from password detail cards.

There are as many cards as indicated on the password count card. Each of them has the following format:

- Col. 1-2 blank
- 3-7 password
- 8 blank
- 9-12 level 1 protection key
- 13 blank
- 14-17 level 2 protection code
- 18 blank
- 19 level 3 protection code

Columns 20-80 are normally blank although columns 20-33 may be utilized in connection with an alternative scheme for level 3 protection.

LEVEL 1 PROTECTION

The level 1 protection key is the sum of the key values for those functions that the user whose password this is can use.

CREATE	0	TOTAL	8
SEARCH	1	INSERT	16
DISPLAY	2	REMOVE	32
END	4	MODIFY	64
		ACCESS	128
		PROTECT	256

These functions are described as follows:

CREATE—is used to create a new data base. It is available to any GIRS user. If a user enters the name of a data base which is not found, GIRS will assume the user wishes to create a new data base and **CREATE** is the only command that will be accepted; conversely, if a specified file is found, the system will disallow the **CREATE** command, making it impossible accidentally to destroy an existing file by its use.

SEARCH—is used to scan parts of the data base for certain character strings. On completion of a search, a summary of successes will be given in the form:

'SEARCH SUCCESSFUL IN RECORD NNNN ...'

Any item which the user is not privileged to see will be ignored in the search summary.

DISPLAY—is used to display information from the data base. Any volume of information can be displayed, from a single element to a complete file. Large volumes of output will ordinarily be run on a batch terminal. All data displays are consistent with the level 2 and level 3 protection codes. Items which the user is not authorized to see are ignored.

END—is used to exit from the system.

TOTAL—is used to aggregate the contents of a defined numeric element over a specified set of records.

The variables are the same as in the **SEARCH** and **DISPLAY** commands. If the element defined is non-numeric, an error response will be generated. A 'blank' option aggregates data over the entire set of records. Level 2 protection is observed although data aggregates are obtained irrespective of level 3 protection codes except where the set of records defined contains six or

less. In this case, a response is generated advising the user to employ the **DISPLAY** command.

INSERT—is used to insert either entire items or entire records.

REMOVE—makes it possible to remove an entire record or an entire item (*i.e.*, one item from each record) from the file. The record number is then free for reuse. When removing an item, the remaining items are renumbered. This renumbering is independent of the level 3 protection code.

MODIFY—is used to change an item in a specified record. The user must specify the item name, record number, and new contents.

ACCESS—is used to change the list of passwords and associated protection keys for the file. Its use will normally be restricted to one or two persons only. It provides two options: to add a new password and its protection keys, or to delete a password. To modify the keys for an existing password it is necessary first to delete the password, and then reinsert it with new keys.

PROTECT—is used to change level 2 and 3 protection in the file; its use should be restricted to one or two users. The command has an **ITEM** option used to change a level 2 protection, and a **RECORD** option used to change a level 3 protection. If the **ITEM** option is omitted, protection is changed for all items in the **RECORD** named.

As an example of the use of level 1 (function) protection: a user privileged only to search a file for statistically aggregated information, that is, *not* permitted to see individually identified information, would have the level 1 code

CREATE	000000000
SEARCH	000000001
END	000000100
TOTAL	000001000
<hr/>	
LEVEL 1 CODE	000001101 = 13

LEVEL 2 PROTECTION

The level 2 protection key is the sum of the level 2 protection codes for those items which the user can access, as given on the item detail cards.

As an example of the use of level 2 (item) protection: a user who is privileged only to work with items 1, 5, and 9 of each record would have the level 2 code:

ITEM 1	0000000001
ITEM 5	0000010000
ITEM 9	0100000000
<hr/>	
LEVEL 2 CODE	0100010001 = 273

Item detail cards comprise a set of one or more cards for each item in the record. The first card of each set contains the following data:

- Col. 1-10 item name
- 11 blank
- 12-13 # lines in the item (1-10)
- 14 blank
- 15-18 level 2 protection code which should be either a power of 2, or 0 for an item which all users can access
- 19 blank
- 20-80 element names (1-10 characters each) separated by backslashes and terminated with a dollar sign (\$) [maximum of 24 element names]

For example, the item detail card for the fourth item of example (b) in Figure 2 would be:

VITA 02 0128 BIRTH DATE\BIRTHPLACE\
 NEXTOF KIN\RELATION\
 ADDRESS \$

LEVEL 3 PROTECTION

The level 3 protection code is set to correspond to the security clearance of the user. He can see items of records whose protection code is less than or equal to the level 3 protection code stored with his password.

Record detail cards contain the actual data in the following format:

- Col. 1-5 record number
- 6 item number (0-9)
- 7 line number within item (0-9)
- 8 level 3 protection code (must be same for all lines in one item). User can access item if his level 3 protection key is \geq this value.
- 9-80 item contents

These cards must be sorted in ascending order on columns 1 to 7, when creating a new data base.

OPTIONAL SYSTEM TO LOCK RECORDS

An additional system for level 3 protection permits locking all items of selected records against certain users. This system is based on the principles of modular algebra and permits mapping a need-to-know security plan into the access system.

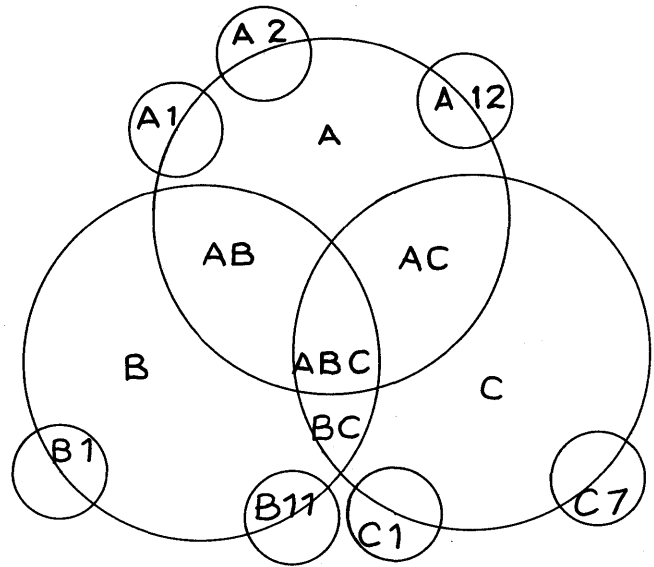


Figure 5—Mapping of a security plan involving three major users with their subusers, and provision for information interchange among major users

Each user is given a pair of divisors (bases) and a pair of remainders (moduli). Every record number the user desires to access is tested for both congruences before the user is privileged to see it.

In the security plan illustrated in Figure 5, there are three major users (A, B, C) which might stand for the accounting, personnel, and production departments of a firm. The records which are the exclusive property of each major user are partitioned into exclusive subsets, which might correspond to: personnel administration, wage and salary administration, training, etc., for the personnel department. In addition, there are intersection or shared-channel sets to facilitate exchange of information between users AB, AC, BC, and all major users.

Columns 20-33 of the password cards can be utilized to store access codes (divisors and remainders) required to access particular sets of records:

- 20 blank
- 21-23 divisor # 1
- 24 blank
- 25-26 remainder # 1
- 27 blank
- 28-30 divisor # 2
- 31 blank
- 32-33 remainder # 2
- 34-80 blank

TABLE I—Partitions of 10,000 Record Numbers

Level 3 Key D ₁ , R ₁ , D ₂ , R ₂	Records Accessible	Record Numbers Assignable	Utilization
2 0 — —	5000	2667	Major user A
3 0 — —	3333	1334	Major user B
5 0 — —	2000	667	Major user C
6 0 — —	1667	1333	Shared channel, users A & B
10 0 — —	1000	667	Shared channel, users A & C
15 0 — —	667	333	Shared channel, users B & C
30 0 — —	333	333	Shared channel, users A, B & C
2 0 13 0	205	205	Subuser A-1
...
2 0 13 12	205	205	Subuser A-13
3 0 11 0	121	121	Subuser B-1
...
3 0 11 10	121	121	Subuser B-11
5 0 7 0	95	95	Subuser C-1
...
5 0 7 6	95	95	Subuser C-7

Table I illustrates the results of partitioning 10,000 record numbers. A record number is selected from a computer-produced list of the members of each set and subset so that the number can be accessed by the major user, subuser, or combination of users who have the specified need to know. Of course, additional passwords can be issued to give various subusers access to partition sets, which can function as common channels for information interchange among them. Note that major user A can access 5,000 records. Out of this set, 2,667 records belong to A exclusively; they cannot be seen by any other major user. User A, however, can allocate these records among his subusers (designated by the alpha-numeric A1 to A13); they are parcelled out at the rate of 205 records per subuser. These records can be seen only by User A and the subuser designated. (Of course, A can hold back a few sets of records for his exclusive use—becoming, actually, his own subuser.)

User A also has the use of 1,333 records; which he shares with User B; only Users A and B can see these records.

User A can also use records taken from the set of 667 records that he shares with C; only Users A and C can see these records.

Finally, User A has the use of records taken from a

common set of 333 records; these records can be seen by all three major Users; A, B, and C.

PROVISION FOR ENCIPHERMENT

There are two points at which this system remains vulnerable to unauthorized entry: a user possessing the general project-programmer pair and password required by the PDP-10 software to access the GIRS system can make use of the peripheral interface program to assign the entire GIRS file to some display device; and a wire-tapper can intercept the transmission of confidential file information to a legitimate user at a remote terminal.

Use of an on-line crypto system can protect the files at these points. The item contents from on record detail cards will be stored in enciphered form for items whose sensitivity requires such precaution. Decipherment can be accomplished at programmable remote terminals; for such items, only enciphered contents will be transmitted over telecommunications lines or be accessible by the peripheral interface program.

A suitable deciphering scheme has been described (1). Essentially it consists of adding modulo 2 to the cipher text stream the bits of the key string used to encipher it. The key string is regenerated and correctly synchronized by using an arithmetic congruential pseudorandom-number generator whose seed string (i.e., ring contents) are produced by a second generator whose operation is specified by a password—which may be the same as used to gain access, or be totally different. In the CRYPTO mode, the output received in answer to a DISPLAY command would be stored on the disk of the programmable terminal—in this case, a PDP-8I.

CONCLUSIONS

This generalized information retrieval system provides a test bed for continuing experimentation with security provisions for multiple-access computer communications systems. By such experimentation, it is anticipated that the optimal trade offs between security and economy can be determined for a wide range of information retrieval applications.

REFERENCES

- 1 J M CARROLL P M McLELLAND
Fast "infinite-key" privacy transformation for resource sharing systems
AFIPS Conference Proceedings FJCC Vol 27 pp 223-230
1970

2 D F BOOTH

File security for a shared file, remote-terminal system
Conf. on Computers: Privacy and Freedom of
Information Queens University Kingston Ontario
May 21-24 1970

3 T D FRIEDMAN

The authorization problem in shared files
IBM Systems Journal Vol 9 No 4 pp 258-280 1970

4 L J HOFFMAN

Computers and privacy: a survey
Computing Surveys Vol 1 No 2 pp 85-103 1969

5 W J STUBGEN M A SHEPHERD

A real-time information editing and retrieval system
Department of Computer Science University of Western
Ontario London Ontario May 1970

Insuring confidentiality of individual records in data storage and retrieval for statistical purposes

by MORRIS H. HANSEN

Westat Research, Inc.
Rockville, Maryland

Much has been written about the question of privacy and the need for the protection of confidentiality of individual records in data storage and retrieval systems. The ability to insure confidentiality is a prime tool in the protection of privacy. The goal of this paper is to summarize from the point of view of a statistician some of the aspects and principles of confidentiality and some of the implications of these principles for computer-based storage and retrieval systems for statistical purposes. The remarks will have special relevance to open retrieval systems, that is, retrieval systems in which customers for information retrieval are the general public, or perhaps specified agencies or groups or individuals, and these customers can retrieve any desired statistics from the confidential records in the files subject to a review to insure that the output conforms to prescribed rules designed to avoid disclosure of individual information. These rules may be concerned with the minimum number of cases on which an individual statistics or frequency count is based or with other aspects, as is discussed later. The access to the data may be restricted to certain authorized types of data through control passwords or keys.

MEANING OF CONFIDENTIALITY

What is meant by confidentiality needs clarification. An obvious meaning is that the individual records, with the names or other identifying information included, will not be made available to other than authorized persons. But beyond this the definition of what is adequate protection of confidentiality needs further clarification.

The Census Bureau has a well-established and well-earned record for preservation of confidentiality of its records. Much of this paper will draw on the Census

Bureau experience as an illustration. With the great concern of the Congress and others over the potential for invasion of privacy in statistical information systems, and especially the proposed and much-discussed federal statistical data center, it is useful to examine how the Census Bureau has come to be widely accepted as a model in the confidentiality protection given to its records. It will be seen that the experience points to serious and as yet unresolved problems, and that the problems are especially difficult for a storage and retrieval system such as a federal data center with access to statistical summaries by persons not authorized to see the individual confidential records.

The Census law (Title 13, U.S.C., Sec. 9-a-2) provides that there shall not be "... any publication [or otherwise make information available] whereby the data furnished by any particular establishment or individual under this title can be identified."

Various interpretations can be made of this language. One is that *no inference* can be made about the results reported by any individual. This is not a tolerable interpretation. At the other extreme, the law cannot reasonably be interpreted to mean that there is no violation of confidentiality provided the name or address (or other specific identification such as Social Security number) is not associated with the information and made available.

A reasonable as distinguished from a rigorous or literal interpretation of the language of the law is required if any statistics are to be published.* For example, the publication of an aggregate of retail sales for hardware stores in a county reveals that no in-

* Here and elsewhere in this paper the term "publication" refers to any means of making information available to persons who do not have authorized access to the confidential records and who are not subject to penalties for disclosure.

dividual hardware store had sales of a greater amount than this aggregate, and this much is revealed about each individual hardware store. Similarly, sometimes the existence (or nonexistence) of an item in each report can be inferred from the publication of statistical aggregates. Thus, the fact that in an age distribution for a specified area from a population census no person is reported as over 75 years of age reveals for each individual person that his age was reported as under 75 years.** In publishing statistics for large areas such considerations may be of little consequence. But in publishing statistics for smaller and smaller areas the problem increases, and the primary role of the decennial census is to produce small area statistics. Especially statistics are needed and produced from the decennial Censuses of Population and Housing for counties, cities, towns, census tracts, and even city blocks within cities or other communities. The storage and retrieval of geographically detailed statistical information may also be a primary goal of other information systems based on a set of administrative records or integrated from administrative systems and perhaps also from statistical surveys.

Years of experience and precedent in publishing statistics by the Bureau of the Census without serious problems suggest the acceptability of the rules and principles that have been followed to avoid unreasonable disclosure of data for individuals in statistical aggregates. However, the computer adds new capabilities as its capacities and applications increase, and these may call for reexamination and some new rules and principles. It is desirable to get recognized, in applying past principles and in developing any new ones, and as has been illustrated in the above discussion, that if *any* statistics are to be published nondisclosure cannot be absolute. Rules for nondisclosure are necessarily based on an interpretation of what is reasonable, and supported by precedents and past experience.

SOME PRINCIPLES AND QUESTIONS FOR GUIDING NONDISCLOSURE IN PROTECTING CONFIDENTIALITY

Some relevant principles or questions concerning rules for protecting confidentiality of individual records will be presented. Clear and unequivocal answers may not exist. Nevertheless, reasonable decisions have been made and must be made, in order to publish census and other statistical results.

** Additional illustrations are presented in 1.

What constitutes protection against exact or approximate disclosure?

Protection against *exact* or *approximate* disclosure of specific items of information in a record must be provided. However, "approximate" disclosure must be interpreted or defined. Issues concerning the approximate disclosure of magnitudes, as distinguished from frequency counts, involve some special considerations.

Illustration: In some studies the Census Bureau has interpreted the disclosure of a magnitude, \times , as not to be an approximate disclosure when the range of interpretation is of the order of $(.75 \text{ to } 1.5) \times$. Frequencies in a distribution may automatically meet this condition if intervals are broad enough for the upper limit of the interval to be at least double the lower limit, as in the following illustration:

Number of employees
Less than 5
5-9
10-24
25-49
50-99
100-199, etc.

Under this rule even an individual case may be reported in such an interval without making an approximate disclosure. Of course the individual is not identified, but frequencies as low as 0, 1, 2 or 3 are shown in such intervals, as in employment size classes for retail stores, by type, within a county, for example, and a person with commonly available local knowledge may be able to identify a particular store identified by a frequency of 1, and its reported employment within the range of the class interval.

Effect of sensitivity of the information

Should disclosure rules take some account of the sensitivity of the information, and be more restricted with highly sensitive information than with less sensitive information? Some information loses sensitivity with time; some may not, or the sensitivity may increase with time. Some information is essentially in the public domain. These factors should, and in fact, do have some impact on the confidentiality treatment, but still without completely specified formal rules.* For

* Alan F. Westin has expressed a need for developing a classification system for personal information to identify types that need various degrees of control. See, for example, *Privacy and Freedom* (Atheneum, 1967).²

example, is there any point in regarding the size of a family or a household (which often is known to everyone in the neighborhood) as equally confidential as the income of the head of the household? Similarly, should the industry code derived from the types of production reported by a manufacturing company be protected as confidential, when often the company spends much money to let the public know of the types of products it makes or the services in which it is engaged? Should the number of employees reported for a plant be protected as equally confidential as the reported sales?

Such questions may have more difficult implications than is readily apparent. Thus, in some instances the number of persons in a household may indicate illegal occupancy to a landlord or to housing code authorities. Again, the industry in which a company is classified may affect the rate of taxation for unemployment compensation. If a company is classified in a high-risk industry instead of a lower-risk one, and if the industry code derived from a confidential statistical report of a company is made public, will it influence the company's tax rate?

Disclosures with supplemental knowledge or collusion

Is it necessary to provide protection against disclosures that can be achieved by collusion, or by supplemental knowledge in addition to one's knowledge of his own affairs? A common rule in avoiding disclosure is that there must be at least three nontrivial cases aggregated in a cell (based on aggregates of magnitudes) so that, for example, a business respondent will not know his competitor's response. Presumably it is not feasible, and the Census Bureau accepts the principle that it is not feasible, to protect against disclosure by collusion. Otherwise, again, nothing could be published. However, the issue of possible disclosure through taking advantage of supplemental knowledge needs further attention, especially in view of the computer capabilities. There is an important difference between analysis to achieve disclosures, with and without the computer. Consider, as an illustration, a cross-tabulation made in great detail.

Assume 10,000 persons in a file for an area, and information for each person on 50 characteristics (something like the results of the questions in a 1970 Population and Housing Census sample questionnaire). Suppose that the record includes some characteristics with two alternative responses, as for sex. Others may have three, five, ten, or twenty alternative classifications (as with ten intervals for an age tabulation). A question

such as occupation may be recorded and tabulated in 100 or many more classes.

If we assume 10 of the questions have 2 alternatives,
10 of the questions have 3 alternatives,
10 of the questions have 5 alternatives,
10 of the questions have 10 alternatives,
10 of the questions have 20 alternatives,

and if we conceive of a cross-tabulation in the fullest possible detail of these 50 questions the number of possible cells becomes $2^{10} \times 3^{10} \times 5^{10} \times 20^{10} \doteq 10^{38}$ cells, which is an astronomical number. It is likely that in such a detailed cross-tabulation each person would be unique, with each cell showing a frequency of zero or 1.

A cross-tabulation of only five of these questions (one from each of the indicated numbers of alternatives) would yield a tabulation with about 6,000 cells, so that a population of 10,000 would have an average of 1.7 per cell in such a tabulation. Of course, many cells may be impossible or blank, and some cells might have several cases. Nevertheless, tabulations in such detail may make it feasible for a person or organization (such as a welfare or taxing agency or a credit bureau) with certain of the same information on some of the people to identify many of them in the tabulation and ascertain other information for them. With a computer the comparison and identification become far more feasible. Consequently, consideration must be given to the amount of detail in which tabulations will be made available in order to preserve confidentiality. Or should and can any possible violations of confidentiality be ignored that can be achieved only through the use of extensive supplemental information? In the computer age this seems unreasonable.

Some interesting discussion and examples of principles and procedures for using collateral information to extract information for individual records from a statistical data bank with retrieval allowed only for statistical aggregates, and by obtaining legitimate responses to queries, are given in an article by Hoffman and Miller.³

The presence of errors, or differences in time reference or in the treatment of individual items of information in two sets of records, is common. Such errors or differences would make more difficult the problem of using collateral information to extract individual information from statistics derived from a set of confidential records used for statistical purposes. However, with sufficiently extensive and detailed independent information available to use in identification, and even in the presence of such errors or differences, the probability of correctly identifying a person

and picking up the desired confidential information increases as the number of cells in a cross-tabulation is increased, or with appropriately designed queries of increasing detail.

Indirect disclosures

Indirect as well as direct disclosures must be considered, and these can be a major source of difficulty. Thus, suppose a small county has six hardware stores, and that a city within the county has four of them. If retail sales are published for the county, and also for the city (we assume each would individually meet disclosure requirements) an indirect disclosure occurs. Each of the two stores in the balance of the county could directly determine his competitor's sales by taking the difference between the county statistics and the city statistics. Thus, if disclosure is to be avoided the data for the city can be made available, and not the county, or for the county and not the city. Indirect disclosures should be avoided, at least in any sensitive type of information.

Priorities need in statistics subject to indirect disclosure restraints

The consequences of indirect disclosures are that priorities are necessary in determining which statistics will be made available and which will not, in order to avoid making available some relatively unimportant information and thereby subsequently denying statistics that have highly important uses. The providing of information forecloses making information available for an alternative, as illustrated above. As another and more serious illustration, it is often true that in the Manufactures or Business Censuses information can be shown for a state total, or for a metropolitan area total, but not for both, and many similar situations arise. Exactly the same kinds of problems can arise in the publications of Population and Housing Census data, especially for small areas where the frequencies get small. In these Censuses, however, some of the data may be less sensitive, and disclosure analysis may not need to be pressed as rigorously. For sensitive data, however, the question becomes: how should one determine the priorities? Obviously, it is public interest and utility that should be determining, but this problem poses many questions beyond the scope of this discussion. Of particular importance, however, is the consequence that the priority problem means that the first comer, who may have a limited use or need in

terms of public interest, may foreclose the possibility of later retrieval of other more important information. The question of priorities adds great complexity to the design of any such retrieval system for information that is subject to confidentiality restraints.

Random modification of data to avoid approximate disclosure

There has been some consideration of random modification of data within the range of, for example, a factor of .5 to 1.5, with the choice of factor within the range made at random, as a means of avoiding approximate disclosure. With this approach an actual report of 850 employees in an establishment might be modified to become $595 = 850(.7)$ where the .7 was chosen at random from the interval .5 to 1.5. The average effect of such modifications on simple aggregates or averages would be relatively small (over a large experience) and numbers so modified in reports can be subjected to less rigorous disclosure rules or even no disclosure analysis. In the case of attributes the approach must be modified to change some fraction of ones to zeros and of zeros to ones, where changes are made at random in ways that do not unduly violate internal consistency of the data for the individual record.

The impact may be more serious with cross-tabulations where the independent variables—those used in sorting into various classes or cells—have been so modified. In this latter case a bias is introduced that may or may not be serious in its magnitude. Such a bias is not necessarily reduced simply by increasing the number of cases within a class.

The random modification of data to avoid approximate disclosure has been considered extensively for various applications in the Bureau of the Census over the past decade or more, but has actually been applied to a very limited extent, so far as I am aware. It has developed and been discussed independently, and again, with limited applications, as a means of preserving confidentiality in retrieval or publication of information.^{4,5} This approach deserves more exploration. It may be that an announced program of random modification of a relatively small fraction of the records selected at random can accomplish much in avoiding disclosure for all of the records in the set.

Disclosure with statistical information from samples

If information in some statistics is based on a sample of a population, the chance of disclosure is reduced, and

the thinner the sample, the less the chance of disclosure in statistical tabulations of a given amount of detail.

For a small enough sampling fraction, even if disclosure rules are not fully observed, the chance of pay-off may be small enough to make prohibitive (as a practical matter) the cost of taking advantage of the potentials for disclosure.

In recognition of this principle the Census Bureau decided to put in the public domain the statistical data recorded for each household from the 1960 Census for a 1 in 1000 sample of households after deleting certain information from the records that would facilitate identification. Of course the name and address were deleted. In addition, geographic identification was deleted below the level of broad city-size class within geographic divisions of the country (there are nine geographic divisions, each consisting of several states). In addition, some extreme cases were modified for sensitive types of information so that, for example, the upper boundary of income reported may have been reduced. Beyond this, the full household information was included in a magnetic tape file on a set of punched cards for the 1/1000 sample, including the housing information and the full listing of individual household members, with the information reported for each individual member. The purpose was to make it feasible for various users to make their own summaries or cross-tabulations or correlations to meet a wide range of needs. It was a great success, with a large number of users of the tape putting it to many uses that could not be served directly by the Census tabulations.

From the point of view of confidentiality, anyone who has a supplemental source of more limited information but that duplicates a number of the items of information in the Census file for individuals or families or households for some part of the population could use that information to identify many of the individual cases in this sample that were also in his file. Of course he could expect to find less than 1 in 1000 of the cases in his file, but for those found he would then have identified the additional information in the 1/1000 file.

Suppose, for example, that a credit bureau had records for a "chunk" of the population in a metropolitan area, including, perhaps, information on age for the head of the family, the number of persons in the family (not necessarily the same as in the household), occupation of the head of the household, whether the home was owned or rented, and the value of the home or the amount of the rent paid. With such information, and even with errors and with differences in time references in both sets, he might run his tape against the Census 1/1000 sample tape, perhaps for a larger

area or areas, and identify with a fairly good chance of success (but with much less than certainty) the cases in the 1/1000 file that were also in his records. He would thereby acquire the additional Census information for the identified cases (including misinformation for cases that were misidentified). But it would cost him a considerable amount both in efforts and dollars, and at the very best he could expect to find a pay-off of less than 1/1000 in the sense of obtaining Census information for the cases in his file. The possibility of misuse arises only in the case of someone with a file of supplemental information that is sufficiently relevant for some subgroup of the population. Even then the pay-off presumably would be very small because of the presence of errors and time reference differences in each source, and the great effort in relation to the number of successful matches (and of course he would not know which of his linked records were the unsuccessful matches). The pay-off might be small not only because of the very small fraction of "finds," but also because the information in the Census records, in general, is not all that sensitive.

Presumably because of such factors no evidence has come to light of any such misuse. At the same time the 1/1000 sample has served many highly useful purposes, so much so that the Census Bureau is proposing to extend the program along the same lines for 1970, and to increase the size of sample from 1/1000 to 1/100.

Disclosure of disclosure rules

There is some thought that rules for disclosure should not be disclosed, and that the availability of the rules will increase the ability of one who wishes to arrive at desired disclosures through analysis of the information that is made available. On this principle, apparently, the Bureau of the Census has not published its various disclosure rules in full, although some of the rules are more or less obvious, and have been made available.

SOME IMPLICATIONS FOR AN OPEN RETRIEVAL SYSTEM

There is need to bring the issues of confidentiality as related to storage and retrieval of information into fuller discussion. The implications of some of the points and principles that have been made above may not be obvious, and study and exploration are needed.

There is no basis for simply assuming that an all-powerful software system can be designed that will take care of the problems of preserving confidentiality in a national statistical data center if one were to be created. Obviously, such a software system cannot be designed until the principles and specific rules of what constitute disclosure and nondisclosure are agreed upon. Unless the principle of *reasonable disclosure*, instead of no disclosure, is adopted, it appears that little or no information could be made available. If the principle of reasonable disclosure is adopted, it will be necessary to define what constitutes reasonable disclosure.

It also must be determined how far the disclosure system will protect against the potentials for disclosure that are made possible by the use of extensive supplemental information acquired through other sources. The availability of such supplemental information can make it feasible to extract increasing amounts of confidential information by making increasingly detailed tabulations or queries, as illustrated earlier, as well as from records such as the 1/1000 sample. Unless the system makes no attempt to protect the disclosure of additional information from sources that have extensive and detailed supplemental information, the disclosure rules may have to be so designed that little of the kinds of anticipated uses from, say, a national statistical data center could be served.

A particularly difficult problem is that of indirect disclosure, through comparisons or analyses of successive tabulations or results of queries. With disclosure analysis that takes account of indirect disclosures many requests might have to be drastically curtailed after a few initial uses. If there were no auditing for indirect disclosure anyone could specify changes in the classifications or specifications for a sequence of tabulations in such a way as to reveal, after analysis, the desired characteristics of many or all of the individual records. Some computer programs have been prepared for dealing with indirect disclosure analysis, and are in use in the Bureau of the Census, but the complexities in a system of open access (subject to restraints on disclosures) seem enormously challenging. A system of recording who has retrieved information, what kinds and how much, for post-audit on a judgmental basis may offer a sufficient protection, especially if a rule of reason is used.

But suppose the problem of indirect disclosure is solved (and in theory, at least, it appears that it can be solved). The problem of priorities still remains. Must all high-priority statistics be listed in advance? Is this feasible? If not, minor or trivial uses of the data may override the subsequent possibility of acquiring information the need for which was not originally foreseen.

One unimportant use may foreclose any possibility of providing information on an urgent and unforeseen current problem.

The issue of priorities is not a new one, as we have seen. It exists in a system in which there is no general access to the stored records. It appears that the problem may be greater in a system that allows access without going into a judgment filter of evaluating public interest and need, or potentials for foreclosing future uses, as is now done in the Bureau of the Census activities. This problem may be a sufficiently serious one to foreclose effective development of anything like a federal statistical data center or data bank that retains confidential records in storage, and permits access by the public or specified groups to statistical tabulations that are audited for disclosure by computer software. The priority problem remains even if other problems prove manageable and can be brought under control.

There is need for fuller discussion of some of these issues by scientific and professional groups. It is not sufficient for these discussions to be conducted separately and in isolation. There is need for interchange using some organized approaches arranged to discuss the issues and problems.

REFERENCES

- 1 P HIRSCH
The world's biggest data bank
Datamation May 1970 pp 66-73
- 2 A F WESTIN
Privacy and freedom
Atheneum New York 1967
- 3 L J HOFFMAN W F MILLER
Getting a personal dossier from a statistical data bank
Datamation May 1970 pp 74-75
- 4 R F BORUCH
Educational research and the confidentiality of data
ACE Research Reports Vol 4 No 4 1969
- 5 R F BORUCH
Maintaining confidentiality of data in educational research: a systemic analysis
American Psychologist Vol 26 No 5 May 1971 pp 413-430
- 6 I P FELLEGI A B SUNTER
A theory for record linkage
Journal of the American Statistical Association Vol 64
No 328 1968 pp 1183-1210
- 7 I P FELLEGI
On the question of statistical confidentiality (unpublished)
Revision of a paper given at the 1970 annual meetings
of the American Statistical Association
- 8 *The computer and invasion of privacy*
Hearings before a Subcommittee of the Committee on
Government Operations House of Representatives
89th Congress Second Session July 26-28 1966

9 C KAYSEN chairman

*Report of the task force on the storage of and access to
government statistics*

Executive Office of the President Bureau of the Budget
October 1966

10 *Privacy and the national data bank concept*

35th Report by the Committee on Government
Operations 90th Congress 2nd Session House Report
No 1842 August 2 1968

11 E V COMBER

Management of confidential information

AFIPS Conference Proceedings Vol 35 1969 Fall Joint
Computer Conference

12 M H HANSEN

Some aspects of confidentiality in information systems
Papers from the Eighth Annual Conference of the Urban
Regional Information Systems Association Louisville
Kentucky September 1970

The formulary model for flexible privacy and access controls*

by LANCE J. HOFFMAN

University of California
Berkeley, California

INTRODUCTION

This paper presents a model for engineering the user interface for large data base systems in order to maintain flexible access controls over sensitive data. The model is independent of both machine and data base structure, and is sufficiently modular to allow cost-effectiveness studies on access mechanisms. Access control is based on sets of procedures called *formularies*. The decision on whether a user can read, write, update, etc., data is controlled by programs (not merely bits or tables of data) which can be completely independent of the contents or location of raw data in the data base.

The decision to grant or deny access can be made in real time at data access time, not only at file creation time as has usually been the case in the past. Indeed the model presented does not make use of the concept of "files," though a specific interpretation of the model may do so. Access control is not restricted to the file level or the record level, although the model permits either of these. If desired, however, access can be controlled at arbitrarily lower levels, even at the bit level. The function of data addressing is separated from the function of access control in the model. Moreover, each element of raw data need appear only once, thus allowing considerable savings in memory and in maintenance effort over previous file-oriented systems.

Specifically not considered in the model are privacy problems associated with communication lines, electromagnetic radiation monitoring, physical security, wire-tapping, equipment failure, operating system software bugs, personnel, or administrative procedures. Cryptographic methods are not dealt with in any detail, though provision is made for inclusion of encrypting

and decrypting operations in any particular interpretation of the model.

Specific interpretations of the model can be implemented on any general-purpose computer; no special time-sharing or other hardware is required. The only *proviso* is that all requests to access the data base must be guaranteed to pass through the data base system.

ACCESS CONTROL METHODS

Access control in existing systems

In most existing file systems which are concerned with information privacy, passwords^{1,2} are used to provide software protection for sensitive data. Password schemes generally permit a small finite number of specific types of access to files. Each file (or user) has an associated password. In order to access information in a file, the user must provide the correct password. These methods, while acceptable for some purposes, can be compromised by wiretapping, electromagnetic radiation monitoring, and other means. Even if this were not the case, there are other reasons³ why password schemes as implemented to date do not solve satisfactorily the problem of access control in a large computer data base shared by many users.

One of these reasons is that passwords have been associated with *files*. In most current systems, information is protected at the file level only—it has been tacitly assumed that all data within a file is of the same sensitivity. The real world does not conform to this assumption. Information from various sources is constantly coming into common data pools, where it can be used by all persons with access to that pool. A problem arises when certain information in a file should be available to some but not all authorized users of the file.

In the MULTICS system⁴ for example, if a user has a file which in part contains sensitive data, he just can-

* Prepared for the U.S. Atomic Energy Commission at the Stanford University Linear Accelerator Center under Contract No. AT(04-3)-515.

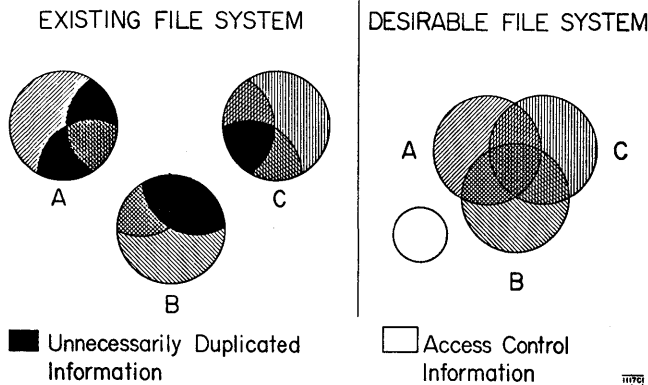


Figure 1—Use of computer storage in file systems

not merge *all* his data with that of his colleagues. He often must separate the sensitive data and save that in a separate file; the common pool of data does not contain this sensitive and possibly highly valuable data. Moreover, he and those he permits to access this sensitive data must, if they also wish to make use of the nonsensitive data, create a distinct merged file, thus duplicating information kept in the system; if some of this duplicated data must later be changed, it must be changed in all files instead of only one. Figure 1, taken from Hoffman's survey⁵ of computers and privacy, graphically illustrates this situation by depicting memory allocation under existing systems and under a more desirable system.

The file management problems presented and the memory wastage (due to duplication of data) tend to inhibit creation of large data bases and to foster the development of smaller, less efficient,* overlapping data bases which could, were the privacy problem really solved, be merged.

Several years ago Bingham⁷ suggested the use of *User's Control Profiles* to associate access control with a user rather than a file. This allows users to operate only on file subsets for which they are authorized and to some extent solves the memory wastage problem. Weissman has recently described a working system at SDC which makes use of security properties of users, terminals, and files.⁸ He presents a set-theoretic model for such a system. His model does not deal with access control below the file level.

Hsiao⁹ has recently implemented a system using *authority items* associated with users. Hsiao's system

controls access at the record level, one step beneath the file level. In it, access control information is stored independently of raw data, and thus can be examined or changed without actually accessing the raw data. Hsiao's system and the TERPS system¹⁰ at West Sussex County in England are two of the first *working* systems which control access at a level lower than the file level.

Access control in proposed systems

Some other methods have been proposed for access control, but not yet implemented. These include Graham's scheme¹¹ which essentially assigns a sensitivity level to each program and data element in the system,** another which allows higher-level programs to grant access privileges to lower-level programs,¹² and still others which place access control at the segment level^{13,14} via machine hardware and "codewords". These methods may prove acceptable in many contexts. However, they are not general enough for all situations. If distinct sensitivity levels cannot be assigned to data, as is sometimes the case, Graham's scheme cannot be used. The other methods, while working in principle on a computer with hardware segmentation, seem infeasible and uneconomical on a computer with another type of memory structure such as an associative memory^{15,16,17,18,19} or a Lesser memory.²⁰ These objections are covered in more detail elsewhere.⁵

Desirable characteristics for an access control method

It seems desirable to devise a method of access control which does not impose an arbitrary constraint (such as segmentation or sensitivity levels) on data or programs. This method should allow efficient control of individual data elements (rather than of files or records only). Also, it should not extract unwarranted cost in storage or elsewhere from the user who wants only a small portion of his data protected. The method should be independent of both machine and file structure, yet flexible enough to allow a particular implementation of it to be efficient. Finally, it should be sufficiently modular to permit cost-effectiveness experiments to be undertaken. We would then finally have a vehicle for exploring the often-asked but never-answered question about privacy controls, "How much does technique X cost?"

We now present such a method.

* A simple cost model for information systems is presented by Arvas.⁶ He there derives a simple rule to determine when it is more efficient to consolidate files and when it is more efficient to distribute copies of them.

** Evidently this scheme has now been implemented.

THE FORMULARY METHOD OF ACCESS CONTROL

We now describe the "formulary" method of access control. Its salient features have been mentioned above. The decision to grant or deny access is made at data access time, rather than at file creation time, as has generally been the case in previous systems. This, together with the fact that the decision is made by a program (not by a scan of bits or a table), allows more flexible control of access. Data-dependent, terminal-dependent, time-dependent, and user response-dependent decisions can now be made dynamically at data request time, in contrast to the predetermined decisions made in previous systems, which are, in fact, subsumed by the formulary method. Access to individual related data items which may have logical addresses very close to each other can be controlled individually. For example, a salary figure might be released without any identification of an employee or any other data.

For any particular interpretation, the installation must supply the procedures listed in Table I. These procedures can all be considered a part of the general accessing mechanism, each performing a specific function. By clearly delimiting these functions, a degree of modularity is gained which enables the installation to experiment with various access control methods to arrive at the modules which best suit its needs for efficiency, economy, flexibility, etc. This modularity also results in access control becoming independent of the remainder of the operating system, a desirable but elusive goal.⁸ While the formulary model and its central ACCESS procedure remain unchanged, each installation can supply and easily change the procedures of Table I as desirable. These procedures are all specified in the body of this paper.

The basic idea behind the formulary method is that a user, a terminal, and a previously built formulary

(defined below) must be linked together, or *attached*, in order for a user to perform information storage, retrieval, and/or manipulative operations. At the time the user requests use of the data base system, this linkage is effected, but only if the combination of user, terminal and formulary is allowed. The general linking process is described later in this section.

Virtual memory mapping hardware is *not* required to implement the model but the model does handle systems equipped with such hardware. It is assumed that enough virtual addressing capacity is available to handle the entire data base. Virtual addresses are mapped into the physical core memory locations, disc tracks, low-usage magnetic tapes, etc., by hardware and/or by the FETCH and STORE primitive operations (see below) for a particular implementation.

Definitions and notation

The *internal name* of a datum is its logical address (with respect to the structure of the data base). The internal name of a datum does not change during continuous system operation.

Examples:

- (1) A "tree name" such as 5.7.3.2 which denotes field 2 of branch 3 of branch 7 of branch 5 in the data base
- (2) "Associative memory identifiers" such as (14, 273, 34) where 14 represents the 14th attribute, 273 represents the 273rd object, and 34 represents the 34th value, in a memory similar to the one described by Rovner and Feldman.²¹

A *User Control Block*, or *UCB*, is space in primary (core) storage allocated during the attachment process (described below). It contains the user identification, terminal identification, and information about the VIRTUAL, CONTROL, SCRAMBLE, and UNSCRAMBLE procedures of the formulary the user is linked to. (An entity with the same name and used similarly has recently been presented independently in a non-implemented model by Friedman.²²)

Usually this information is just the virtual address of each of these procedures. The virtual addresses are kept in primary storage in the UCB since a formulary, once linked to a user and terminal, will probably be (oft-) used very shortly. The first reference to any of these addresses (indirectly through the UCB) will trigger an appropriate action (e.g., a page fault on some computers) to move the proper program into primary storage (if it is not there already). It will then presumably stay there as long as it is useful enough to

TABLE I—Procedures Supplied by the Installation

FOR EACH INTERPRETATION, INSTALLATION MUST SUPPLY

- AT LEAST ONE TALK PROCEDURE
- CODING FOR THE ACCESS ALGORITHM
- PRIMITIVE OPERATIONS
 - FETCH
 - STORE
- AT LEAST ONE FORMULARY, CONSISTING OF
 - CONTROL PROCEDURE
 - VIRTUAL PROCEDURE
 - SCRAMBLE PROCEDURE (may be null)
 - UNSCRAMBLE PROCEDURE (may be null)
- A FORMULARYBUILDER PROCEDURE

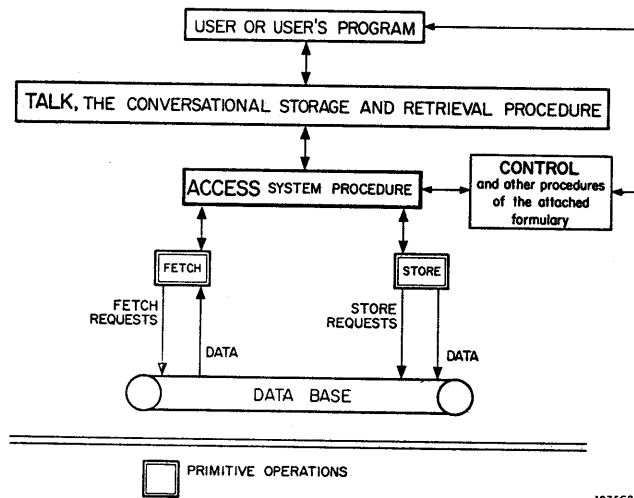


Figure 2—User/data base interface

merit keeping in high-speed memory. The virtual addresses of procedures of a formulary cannot change while they are contained in any UCB. This constraint is easy to enforce using the CONTROL procedure described below which controls operations on any datums, including formularies. Each UCB always is in high-speed primary storage in the data area of the ACCESS procedure.

The ACCESS procedure

All control mechanisms in the formulary model are invoked by a central ACCESS procedure. This ACCESS procedure is the only procedure which directly calls the primitive FETCH and STORE operations and which performs locking and unlocking operations on data items in the base. All requests for operations on the data base must go through the ACCESS procedure.

The ACCESS procedure is a very important element of the formulary model. It is described in full detail and its algorithm is supplied below.

The user communicates only indirectly with ACCESS. The bridge (see Figure 2) between the system-oriented ACCESS procedure and the application-oriented user is provided by the (batch or conversation) storage and retrieval program, TALK.

TALK, the application-oriented storage and retrieval procedure

To access a datum, the user must call upon TALK, the (nonsystem) application oriented storage and re-

trieval procedure. TALK converses with the user (or the user's program) to obtain, along with other information, (1) a datum description in a user-oriented language, and (2) the operation the user wishes to perform on that datum. TALK translates the datum description in the user-oriented language into an internal name, thus providing a bridge between the user's conception of the data base and the system's conception of the data base. The TALK procedure is described in more detail below.

Formularies—what they are

A formulary is a set of procedures which controls access to information in a data base. These procedures are invoked whenever access to data is requested. They perform various functions in the storage, retrieval, and manipulation of information. The set of procedures and their associated functions are the essential elements of the formulary model of access control.

Different users will want different algorithms to carry out these functions. For example, some users will be using data which is inaccessible to others; the name of a particular data element may be specified in different ways by different users; some users will manipulate data structures—such as trees, lists, sparse files, ring structures, arrays, etc.—which are accessed by algorithms specifically designed for these structures. Depending on how he wishes to name, access, and control access to elements of the data base, each user will be attached to a formulary appropriate to his own needs.

Procedures of a formulary

In this subsection, we describe the procedures of a formulary. These procedures determine the accessibility, addressing, structure and interrelationships of data in the data base dynamically, at data request time. They can be arbitrarily complex. In contrast, earlier systems usually made only table-driven static determinations, prespecified at file creation time.

Each procedure of a formulary should, if possible, run from execute-only memory, which is alterable only under administrative control. The integrity of the system depends on the integrity of the formularies and therefore the procedures of all formularies should be written by "system" programmers who are assumed honest. These procedures should be audited for program errors, hidden "trap doors," etc., before being inserted into the (effective) execute-only memory under administrative control. Failure to do this may result in the compromising of sensitive data, since an unscrupu-

lous programmer of a formulary could cause the formulary to "leak" sensitive information to himself or to his agents.

A formulary has four procedures: VIRTUAL, SCRAMBLE, UNSCRAMBLE, and CONTROL. The first three are relevant but not central to access control; the decision on whether to grant the type of access desired is made solely by the CONTROL procedure. The first three procedures are explicitly included in each formulary for three reasons:

- (1) to centralize in one place all functions dealing with addressing and access control;
- (2) to give the model the generality necessary to model existing and proposed systems; and
- (3) to provide well-delimited modules for cost/effectiveness studies and for experimentation with different addressing schemes and access control schemes.

a. The VIRTUAL procedure. VIRTUAL translates an internal name into the virtual address of the corresponding datum. VIRTUAL is a procedure with two input parameters:

- (1) the internal name to be translated
- (2) a cell which will sometimes be used to hold "other information" as described below.

VIRTUAL returns

- (1) the resulting virtual address
- (2) a completion code (1 if normal completion)

Recall that enough virtual addressing capacity is assumed available to handle the entire data base. Virtual addresses are mapped into the physical core memory locations, disc tracks, low-usage magnetic tapes, etc., by hardware and/or by the FETCH and STORE primitive operations for a particular implementation.

b. The SCRAMBLE procedure. SCRAMBLE is a procedure which transforms raw data into encrypted form. (In some specific systems, SCRAMBLE may be null.) SCRAMBLE has two input parameters:

- (1) the virtual address of the datum to be scrambled
- (2) the length of the datum to be scrambled

SCRAMBLE has three output parameters:

- (1) a completion code (1 if normal completion)
- (2) the virtual address of the scrambled datum
- (3) the length of the scrambled datum

Note that if an auto-key cipher (one which must access the start of the cipher-text, whether or not the information desired is at the start) is used, *all* of the information encrypted using that cipher, be it as small as a single field or as large as an entire "file," *must* be governed by the same access control privileges. Therefore, some applications may choose to use several (or many) auto-key ciphers within the same "file." It is inefficient and usually undesirable to scramble data items at other than the internal name level, e.g., scrambling as a block (to effectively increase key length) the data represented by several internal names. In cases where internal names represent data which fit into very small areas of storage, greater security may be obtained by other methods (e.g., use of nulls).

We do not discuss encrypting schemes in this paper. The interested reader is referred to work by Shannon,²³ Kahn,²⁴ and Skatrud.²⁵

c. The UNSCRAMBLE procedure. UNSCRAMBLE is an unscrambling procedure which transforms encrypted data into raw form. (In some specific systems, UNSCRAMBLE may be null.) UNSCRAMBLE has two input parameters:

- (1) the virtual address of the datum to be unscrambled
- (2) the length of the datum to be unscrambled

UNSCRAMBLE has three output parameters:

- (1) a completion code (1 if normal completion)
- (2) the virtual address of the unscrambled datum
- (3) the length of the unscrambled datum.

d. The CONTROL procedure. CONTROL is a procedure which decides whether a user is allowed to perform the operation he requests (FETCH, STORE, FETCHLOCK, etc.) on the particular datum he has specified. CONTROL may consider the identification of the user and/or the source of the request (e.g., the terminal identification) in order to arrive at a decision. CONTROL may also converse with the requesting user before making the decision.

CONTROL has two input parameters and two output parameters. The two input parameters are:

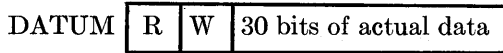
- (1) the internal name of the datum
- (2) the operation the user desires to perform

The two output parameters are:

- (1) 1 if access is allowed; otherwise an integer greater than 1
- (2) "other information" (explained below).

In some specific systems, data elements may themselves contain access control information. Consider three examples:

Example 1.



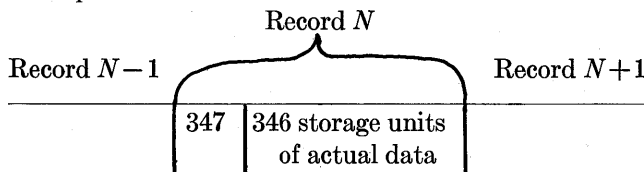
If bit R is on, DATUM is readable.
If bit W is on, DATUM is writeable.

Example 2.



Reading or writing of salaries of \$25,000 or over requires special checking. CONTROL must inspect the SALARY cell before it can do further capability checking and eventually return 1 or some greater integer as its first output parameter (see Figure 3). Note that return of an integer greater than 1 actually transmits some information to the user; if he knows that he will not be allowed to alter salaries which are \$25,000 or over, a denial of access actually tells him that the salary in question is at least \$25,000. In the formulary model, CONTROL can only make a yes or no decision about access to a particular datum. Any more complex decisions, such as one involving release of a count which is possibly low enough to allow unwanted identification of individual data²⁶ (e.g., "Tell me how many people the Health Physics Group treated for radiation sicknesses last year who also were treated by the Psychiatric Outpatient Department at the hospital"), can only be made by a suitably sophisticated TALK procedure.

Example 3.



The record contains its own length (and, therefore, also points to its successor). This type of record would appear, for example, in variable length sequential records on magnetic tape and in some list-processing applications.

In systems of this type, CONTROL might often duplicate VIRTUAL's function of transforming the internal name of a datum into that datum's virtual address. To achieve greater efficiency, CONTROL can (when appropriate) return the datum's virtual address as "other information." VIRTUAL, which is called after CONTROL (see the ACCESS algorithm below),

can then examine this "other information." If a virtual address has been put there by CONTROL, VIRTUAL will not duplicate the possibly laborious determination of the datum's virtual address, since this has already been done. VIRTUAL will merely pluck the address out of the "other information" and pass it back.

Note that CONTROL can be as sophisticated a procedure as desired; it need not be merely a table-searching algorithm. Because of this, CONTROL can consider many heretofore ignored factors in making its decision (see Figure 3). For example, it can make decisions which are data-dependent and time-dependent. It can require two keys (or *N* keys) to open a lock. Also it can carry on a lengthy dialogue with the user before allowing (or denying) the access requested.

CONTROL is not limited to use at data request time. In addition to being used to monitor the interactive storage, retrieval, and manipulation of data, it can also be used at initial data base creation time for data edit picture format checking, data value validity checking, etc. Or, alternatively, one could have two procedures CONTROL1 and CONTROL2, in two different formularies, F1 and F2. F1 could be attached at data input time and F2 at on-line storage, retrieval, manipulation, and modification time.

Simultaneous use of one formulary by multiple users

Note that the same formulary can be used simultaneously by several different users with different access permissions. This is possible because access control is determined by the CONTROL procedure of the attached formulary. This procedure can grant different privileges to different users.

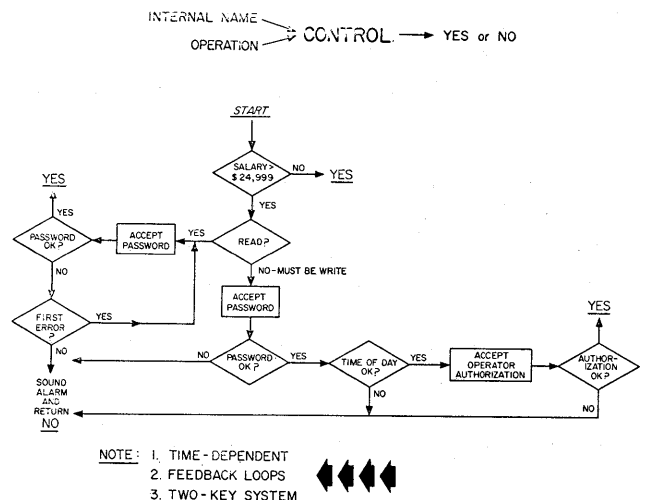


Figure 3—A sample CONTROL procedure

Building a formulary

Before a formulary can be attached to a user and a terminal, the procedures it contains must be specified. This is done using the system program FORMULARY-BUILDER. FORMULARYBUILDER converses with the systems programmer who is building a formulary to learn what these procedures are, and then retrieves them from the system library and enters them as a set into a formulary which the user names. The specifics of FORMULARYBUILDER depend on the particular system.***

The attachment process—the method of linking a formulary to a user and terminal

In order to allow information storage and retrieval operations on the data base to take place, a user, a terminal, and a formulary which has been previously built using FORMULARYBUILDER must be linked together. This linking process is done in the following manner.

At the first time ACCESS is called (by TALK) for a given user and terminal, it will only permit attachment of a formulary to the user and terminal (i.e., it will not honor a request to fetch, store, etc.). The attachment is permitted only if the CONTROL program of the default formulary allows. The default formulary, like all other formularies, contains VIRTUAL, CONTROL, SCRAMBLE, and UNSCRAMBLE procedures. For the default formulary, they act as follows:

CONTROL	CONTROL takes the internal name representing the formulary and decides whether user U at terminal T is allowed to attach the formulary represented by the internal name. U and T are maintained in the UCB and passed to CONTROL by ACCESS.
VIRTUAL	VIRTUAL takes the internal name representing the formulary and returns the virtual address of the formulary.
SCRAMBLE	No operation.
UNSCRAMBLE	No operation.

The ATTACH attempt, if successful, causes informa-

tion about the formulary specified by the user to be read into the UCB (which is located in the data area of the ACCESS procedure). ACCESS then uses this information (when it is subsequently called on behalf of this user/terminal combination) to determine which CONTROL, VIRTUAL, SCRAMBLE, and UNSCRAMBLE procedures to invoke.

Independence of addressing and access control

After the attachment process, the User Control Block (UCB) contains the user identification U , terminal identification T , and information about (usually pointers to) the VIRTUAL, CONTROL, SCRAMBLE, and UNSCRAMBLE procedures of a formulary. Whether the user can perform certain operations on a given datum is controlled by the CONTROL program. The addressing of each datum is controlled by the VIRTUAL program. Addressing of data items is now completely independent of the access control for the data items.

Breaking an attachment

An existing attachment is broken whenever

- (1) the user indicates that he is finished using the information storage and retrieval system (either by explicitly declaring so or implicitly by logging out, removing a physical terminal key, reaching the end-of-job indicator in his input card deck, etc.), or
- (2) the user, via his TALK program, explicitly detaches himself from a formulary.

Subdivision of data base into files not required

Note that while the concept of a data set (or a "file") MAY be used, the formulary method does not require this. This represents a significant departure from previous large-scale data base systems which were nearly all organized with files (data sets) as their major subdivisions. Under the formulary scheme, access to information in a data set is not governed by the data set name. Rather, it is governed by the CONTROL procedure of the attached formulary. Similarly, addressing of data in a data set is governed by the VIRTUAL procedure and not by the data set name. Subdividing a data base into data sets, while certainly permitted and often desirable, is not required by the formulary model.

*** An extension to FORMULARYBUILDER which would allow a user to grant capabilities to other users, and then allow these users to grant capabilities to still other users, etc., has been proposed by Victor Lesser. The formulary model does not currently adequately handle this area of concern.

Concurrent requests to access data—the LOCKLIST

The problem of two or more concurrent requests for exclusive data access necessitates a mechanism to control these conflicts among competing users. This problem has been discussed and solutions proposed by several workers.^{28,9,27} In the formulary model, data can be set aside (locked) dynamically for the sole use of one user/terminal combination in a manner similar to Hsiao's "blocking"⁹ using a mechanism known as the LOCKLIST.

The locking and unlocking of data to control simultaneous updating is an entirely separate function from the access control function. Access control takes into account privacy considerations only. Locking and unlocking are handled by a separate mechanism, the LOCKLIST. This is a list of triplets maintained by the ACCESS program and manipulated by the FETCHLOCK, STORELOCK, UNLOCKFETCH, and UNLOCKSTORE operations. Each triplet contains (1) the internal name of a current item, (2) the identification of the user/terminal combination which caused it to be locked, and (3) the type of lock (fetch or store). Any datum represented by a triplet on the LOCKLIST can be accessed only by the user/terminal combination which caused it to be locked.

Data items which can be locked are atomic, i.e., subparts of these data items *cannot* be locked. This implies, for example, that if a user wishes to lock a tree structure and then manipulate the tree without fear of some other user changing a subnode of the tree, either

- (1) the tree must be atomic in the sense that its subnodes do not have internal names in the data base system, or
- (2) each subnode must be explicitly locked by the user and only after all of these are locked can he proceed without fear of another user changing the tree. ****

The TALK procedure—details

To access a datum, the user *must* effectively call upon TALK, the (nonsystem) application-oriented storage and retrieval procedure. TALK converses with the interactive user and/or the user's program and/or

**** A more general and elegant method of handling concurrent requests to access data is being developed by R. D. Russell as part of a general resource allocation method. Much of the housekeeping work currently done in the formulary model can be handled by his method.

the operating system to obtain

- (1) a datum description in a user-oriented language
- (2) the operation the user wishes to perform on that datum
- (3) user identification and other information about the user and/or the terminal where the user is located.

Depending on the particular system, the user explicitly gives TALK zero, one two, or all three of the above parameters. TALK supplies the missing parameters (if any), converts (1) to an internal name, and then passes the user identification, the terminal identification, the internal name of the datum, and the desired operation to the ACCESS procedure, which actually attempts to perform the operation.

Note that one system may have available many TALK procedures. A user requests invocation of any of them in the same way he initiates any (non-system) program. Sophisticated users will require only "bare-bones" TALK procedures, while novices may require quite complex tutorial TALK procedures. They may both be using the same data base while availing themselves of different datum descriptions. As an example, one TALK procedure might translate English "field names" into internal names, while another TALK procedure translates French "field names" into internal names. This ability to use multiple and user-dependent descriptions of the same item is not available with such generality in any system the author is aware of, though some systems allow lesser degrees of this.^{29,30}

Different TALK procedures also allow concealment of the fact that certain information is even in a data base, as illustrated in Figure 4. The remarks above

<p><u>USER 1</u></p> <p>WHAT PROGRAM? talk1 TALK1 HAS BEGUN EXECUTION. WHAT DATA WOULD YOU LIKE TO SEE? salary of robert d. jones YOU ARE NOT PERMITTED READ ACCESS TO THE <u>SALARY</u> FIELD.</p>	<p><u>USER 2</u></p> <p>WHAT PROGRAM? talk2 TALK2 HAS BEGUN EXECUTION. WHAT DATA WOULD YOU LIKE TO SEE? salary of robert d. jones NO FIELD NAMED <u>SALARY</u>.</p>
<p>CONTROL determined that the user was not permitted read access, causing this reply to be given by TALK1.</p>	<p>TALK2 intentionally returned this reply to the user.</p>

Figure 4—Concealment of the fact that a data base contains certain information

about using different TALK procedures also apply if a system uses only one relatively sophisticated TALK procedure which takes actions dependent on the person or terminal using it at a given time.

The ACCESS procedure—details

ACCESS uses the VIRTUAL, CONTROL, UNSCRAMBLE, and SCRAMBLE procedures specified in the UCB to carry out information storage and retrieval functions. Its input parameters are:

- (1) information about the user, terminal, etc., defined by the installation. This information is passed by the procedure that calls ACCESS;
- (2) internal name of datum;
- (3) an area which either contains or will contain the value of the datum specified by (2);
- (4) the length of (3);
- (5) operation to perform—FETCH, FETCHLOCK, STORE, STORELOCK, UNLOCKFETCH, UNLOCKSTORE, ATTACH, or DETACH. FETCHLOCK and STORELOCK lock datums to further fetch or store accesses respectively (except by the user/terminal combination for which the lock was put on). UNLOCKFETCH and UNLOCKSTORE unlock these locks. ATTACH and DETACH respectively create and destroy user/terminal/formulary attachments.
- (6) a variable in which a completion code is returned by ACCESS.

ACCESS itself handles all operations of (5) except FETCH and STORE. For FETCH and STORE operations on the data base, it invokes the FETCH and STORE primitives specified below.

Note that some means must be provided to determine which formulary is attached so the CONTROL, SCRAMBLE, UNSCRAMBLE, and VIRTUAL procedures of that particular formulary can be invoked. One method is to have those procedures themselves determine which formulary is attached by examining data common to them and to the ACCESS procedure. These data are initially set by the ACCESS procedure and then are referenced by the other procedures. A working system using this method is illustrated in another report.³¹ An alternative method, if ACCESS is written in a more powerful language or in assembly language, would be to use a common transfer vector.

Note that the procedures TESTANDSET and IDXLL and their corresponding calls can be removed from ACCESS if no user will ever have to lock out access to a datum which ordinarily can be accessed by several users at the same time or if the installation wishes to use another method to control conflicts among users competing for exclusive access to datums; this makes the procedure considerably shorter. Such a "no parallelism" version of the ACCESS algorithm is given elsewhere.³¹

An ALGOL algorithm for the ACCESS procedure follows. This procedure is quite important and should be examined carefully. The comments in the algorithm should not be skipped, as they often suggest alternate methods for accomplishing the same goals.

THE ACCESS ALGORITHM

procedure access (info, intname, val, length, opn, compcode);

integer array info, val; **integer** length, opn, compcode;

begin comment If OPN = FETCH, VAL is set to the value of the datum represented by INTNAME.

If OPN = STORE, the value of the datum represented by INTNAME is replaced by the value in the VAL array.

If OPN = FETCHLOCK or STORELOCK, the datum is locked to subsequent FETCH or STORE operations by other users or from other terminals until an UNLOCKFETCH or UNLOCKSTORE operation, whichever is appropriate, is performed.

If OPN = UNLOCKFETCH or UNLOCKSTORE, the fetch lock or store lock previously inserted by a FETCHLOCK or STORELOCK operation is removed.

If OPN = ATTACH, the formulary represented by internal name INTNAME is attached to the user and terminal described in the INFO array.

If OPN = DETACH, the formulary represented by internal name INTNAME is detached from the user and terminal described in the INFO array.

VAL is LENGTH storage elements long.

Note that a FETCH (STORE) operation will actually attempt to fetch (store) LENGTH storage elements of information.

It is the responsibility of the TALK procedure to handle scrambling or unscrambling algorithms that return outputs of a different length than their inputs.
ACCESS returns the following integer completion codes in COMPCODE:

- 1 normal exit, no error
- 2 unlock operation requested by user or terminal who/which did not set lock
- 3 operation permitted but gave error when attempted
- 4 attempt to unlock datum which is not locked in given manner
- 5 cannot handle any more User Control Blocks (would cause table overflow)
- 6 attempt to detach nonexistent user/terminal/formulary combination
- 7 operation permitted for this user and terminal but could not be carried out since datum was locked (by another user/terminal) to prevent such an operation
- 8 cannot put lock on as requested since LOCKLIST is full
- 9 datum already locked by this user and terminal
- 10 error return from VIRTUAL procedure
- 11 operation on the datum represented by INTNAME not permitted by CONTROL procedure of the attached formulary
- 12 end of data set encountered by FETCH operation.

Note that by the time the user has left the ACCESS routine, the data may have been changed by another user (if the original user did not lock it). Note that ACCESS could be altered to allow scrambling and unscrambling to take place at external devices rather than in the central processor.

Important: ACCESS expects the following to be available to it. The installation supplies these in some way other than as parameters to ACCESS (for example, as global variables in ALGOL or COMMON variables in FORTRAN)--

- (1) **ISTDUCB** the default User Control Block. Its length is **NUCB** storage units.
- (2) **NUCB** see (1).
- (3) **UCB** a list of User Control Blocks (UCB's) initialized outside ACCESS to $ucb(1, 1) = -2$,
 $ucb(i, j) = \text{anything when } \sim(i = j = 1)$
UCB is declared as **integer array** [1:maxusers, 1:nucb].
- (4) **MAXUSERS** the maximum number of users which can be actively connected to the system at any point in time.
- (5) **ITALK** the length of the **INFO** array (which is the first parameter of ACCESS)--**INFO** contains information about the user and terminal which is used by ACCESS and also passed by ACCESS to procedures of the attached formulary.
INFO[1] contains user identification.
- (6) **LOCKLIST** a list of locks (each element of the **LOCKLIST** array should be initialized outside ACCESS to -1) **LOCKLIST** is declared as **integer array** [1:4, 1:maxlist].
- (7) **MAXLLIST** the maximum length of the **LOCKLIST**.
- (8) **CS1** a semaphore to govern simultaneous access to the critical section of the ACCESS procedure (initialized to 1 outside ACCESS).

ACCESS assumes that the variables **FETCH**, **STORE**, **FETCHLOCK**, **STORELOCK**, **UNLOCKFETCH**, **UNLOCKSTORE**, **ATTACH**, and **DETACH** have been initialized globally and are never changed by the installation;

integer array iucb [1:nucb], reslt [1:length];

integer i, ii, islot, j, yesno, other, n, datum;

integer procedure testandset (semaphore); **integer** semaphore;

begin comment TESTANDSET is an integer function designator. It returns -1 if SEMAPHORE was in the state LOCKED on entry to TESTANDSET. Otherwise, TESTANDSET returns something other than -1. In all cases, SEMAPHORE is in state LOCKED after the execution of the TESTANDSET procedure, and must be explicitly unlocked in order for it to be used again.

TESTANDSET is used to implement a controlling mechanism to prevent conflicts among users competing for the same resource, as discussed in work by Dijkstra.²⁷ It will NOT prevent “deadly embraces”.³² No explicit code is given here, since the function is machine-dependent.

This procedure can be removed if no user will ever have to lock out access to a datum which ordinarily can be accessed by several users at the same time or if the installation wishes to use another method to control conflicts among users competing for exclusive access to datums;

<code>

end testandset;

integer procedure idxll (intname, opn); **integer** intname, opn;

begin comment IDXLL, given an internal name INTNAME, returns the relative position of INTNAME on the LOCKLIST if the datum represented by INTNAME is locked in a manner affecting the operation OPN. Otherwise, IDXLL returns the negation of the relative location of the first empty slot on the LOCKLIST. If the LOCKLIST is full and the INTNAME/OPN combination is not found on it, IDXLL returns 0.

This procedure can be removed if no user will ever have to lock out access to a datum which ordinarily can be accessed by several users at the same time or if the installation wishes to use another method to control conflicts among users competing for exclusive access to datums;

integer firstempty;

j := if opn = FETCH **or** opn = UNLOCKFETCH **or** opn = FETCHLOCK **then** 1 **else** 2;

idxll := firstempty := 0;

for i := 1 **step** 1 **until** maxllist **do**

begin ii := -i;

if locklist [1, i] = -1 **then** firstempty := i

else if locklist [1, i] = intname **and** locklist [2, i] = j **then begin** idxll := i;

go to RET

end;

end;

if firstempty \neq 0 **then** idxll := -firstempty;

RET:

end idxll;

procedure ret (i); **integer** i;

begin comment RET sets the completion code compcode to i and then causes exit from the ACCESS procedure;

compcode := i; **go to** FIN

end ret;

compcode := 1;

comment first let's see if we recognize the user/terminal combination in INFO;

islot := 0;

for i := 1 **step** 1 **until** maxusers **do**

begin ii := i;

if ucb [i, 1] = -2 **then begin comment** end of list of ucb's;

if islot = 0 **then begin if** ii \neq maxusers **then** ucb [ii + 1, 1] := -2;

go to XFER;

end

else go to PRESETUP;

end

else if ucb [i, 1] = -1 **then** islot := ii

comment remember this slot if vacant;

else begin for j := 1 **step** 1 **until** italk **do**

if ucb [i, j] \neq info [j] **then go to** ILOOPND;

go to SETUPPTRS

end;

ILOOPND:

end i loop;

if islot = 0 **then** ret (5); **comment** cannot handle any more UCBs;

PRESETUP:

ii := islot;

XFER:

for k := 1 **step** 1 **until** italk **do** ucb [ii, k] := info [k];

for k := italk + 1 **step** 1 **until** nucb **do** ucb [ii, k] := istdub [k];

SETUPPTRS:

for i := 1 **step** 1 **until** nucb **do** iucb [i] := ucb [ii, i];

comment set up pointers to appropriate user control block for particular implementation. Note well: Setting up pointers to appropriate user control blocks is quite dependent on the particular system;

comment We have now associated user and terminal with the user control block (representing a formulary) in relative position i of the UCB table;

if iucb [nucb] \neq inname **and** opn = DETACH **then** ret (6);

comment attempt to detach user/terminal/formulary combination not currently attached;
control (inname, opn, yesno, other);

if yesno > 1 **then** ret (11);

comment return 11 if CONTROL does not permit operation;

if opn = ATTACH **then begin** ucb [ii, nucb] := inname; **go to** FIN
 end;

comment Note well: In many implementations, pointers to each procedure of the formulary (obtained by having VIRTUAL transform inname into a virtual address) might be put into the UCB upon attachment. In other, the philosophy used here of only putting one pointer--to the formulary--into the UCB will be followed. The decision should take into account design parameters such as implementation language, storage available, etc.;

if opn = DETACH **then begin comment** detach formulary (this leaves an open slot in the ucb array);
 ucb [ii, 1] := -1; **go to** FIN
 end;

if opn = UNLOCKFETCH **or** opn = UNLOCKSTORE **then**

begin i := idxll (inname, opn); **comment** find internal name on LOCKLIST;

if i \leq 0 **then** ret (4); **comment** cannot find it;

for j := 1 **step** 1 **until** italk **do**

if locklist [2 + j, i] \neq iucb [j] **then** ret (2);

 locklist [1, i] := -1; **comment** undo the lock and mark slot in UCB array empty;

go to FIN

end unlock operation;

TRY:

if testandset (cs1) = -1 **then go to** TRY;

comment loop until no other user is executing the critical section below;

comment ACCESS should ask to be put to sleep if embedding system permits;

comment-----enter critical section for locking out datums-----;

i := idxll (inname, opn);

comment get relative location of locked datum in locklist;

if i > 0 **then begin comment** datum found on locklist so see if it was locked by this user and terminal;

for j := 1 **step** 1 **until** italk **do**

if locklist [2 + j, i] \neq iucb [j] **then** ret (7);

comment data already locked by another user or terminal;

if opn = FETCHLOCK **or** opn = STORELOCK **then** ret (9);

comment datum already locked by this user and terminal, so return completion code of 9;

end;

i := -i;

if opn = FETCHLOCK **or** opn = STORELOCK **then**

begin comment this is a lock operation;

```

if i = 0 then ret (8); comment cannot set lock since locklist is full;
locklist [2, i] := if opn = FETCHLOCK then 1 else 2;
comment set appropriate lock;
for j := 1 step 1 until italk do locklist [2 + j, i] := iucb [j];
comment place user and terminal identification into LOCKLIST;
locklist [1, i] := intname; comment place internal name on LOCKLIST;
go to FIN;
end lock operation;
virtual (intname, datum, other, compcode);
comment VIRTUAL returns in datum the virtual address of the datum specified;
if compcode > 1 then ret (10); comment error return from VIRTUAL;
if opn = STORE then
  begin comment store operation;
  scramble (val, length, compcode, reslt, n);
  if compcode > 1 then ret (3);
  comment operation permitted but gave error when attempted;
  comment now perform a physical write of n storage units to the block starting at reslt;
  store (datum, reslt, n, compcode);
  if compcode > 1 then ret (3)
  end
else
  begin comment fetch operation;
  fetch (datum, reslt, length, compcode);
  if compcode = 2 then ret (12); comment end of data set encountered;
  if compcode > 1 then ret (3);
  unscramble (reslt, length, compcode, val, n);
  if compcode > 1 then ret (3);
  end fetch operation;
FIN:
comment-----Leave critical section for locking out datums-----;
csl := 1;
end access;

```

FETCH and STORE primitive operations

The two primitive operations FETCH and STORE are supplied by the installation. These primitives actually perform the physical reads and writes which cause information transfer between the media the data base resides on and the primary storage medium (usually, magnetic core storage). They are invoked only by the ACCESS procedure.

The primitive operations cannot be expressed in machine-independent form, but rather depend on the specific system and machine used. They are defined functionally below.

FETCH (ADDR, VALUE, LENGTH, COMP)

This primitive fetches the value which is contained in the storage locations starting at virtual address ADDR and returns it in VALUE. This value may be

scrambled, but if so unscrambling will be done later by UNSCRAMBLE (called from ACCESS), and LENGTH is the length of the *scrambled* data. The value comprises LENGTH storage elements. Upon completion, the completion code COMP is set to:

- 1 if normal exit
- 2 if end of data set encountered when physical read attempted
- 3 if length too big (installation-determined)
- 4 if illegal virtual address given to fetch from
- 5 if error occurred upon attempt to do physical read.

STORE (ADDR, VALUE, LENGTH, COMP)

This primitive stores LENGTH storage elements starting at virtual address VALUE into LENGTH storage elements starting at virtual address ADDR.

The information stored may be scrambled, but if so the scrambling has already been done by SCRAMBLE (called from ACCESS), and LENGTH is the length of the *scrambled* data. Upon completion, the completion code COMP is set to:

- 1 if normal exit
- 3 if length too big (installation-determined)
- 4 if illegal virtual address given to store into
- 5 if error occurred upon attempt to do physical write.

A NOTE ON THE COST OF SOME PRIVACY SAFEGUARDS

As mentioned above, a desirable property for an access control model is that it be sufficiently modular to permit cost-effectiveness experiments to be undertaken. In this way the model would serve as a vehicle for exploring questions of cost with respect to various privacy safeguards.

Using the formulary model, an experiment was run on the IBM 360/91 computer system at the SLAC Facility of Stanford University Computation Center. This experiment was designed to obtain figures on the additional overhead due to using the formulary method and on the costs on encoding (and conversely the cost of decoding data). Early results³¹ seem to indicate that the incremental cost of scrambling information in a large computer data base where fetch accesses (and hence unscrambling operations) are relatively infrequent is infinitesimal.

It is easy to use the formulary model to carry out various other experiments dealing with relative costs of diverse encoding methods and data accessing schemes. We hope to do more of this in the future.

SUMMARY

We have defined and demonstrated a model of access control which allows real-time decisions to be made about privileges granted to users of a data base. Raw data need appear only once in the data base and arbitrarily complex access control programs can be associated with arbitrarily small fragments of this data.

The desirable characteristics for an access control method laid out in the section on access control methods are all present (though we have not yet run enough experiments to make general statements about efficiency):

- (1) No arbitrary constraint (such as segmentation

or sensitivity levels) is imposed on data or programs.

- (2) The method allows control of individual data elements. Its efficiency depends on the specific system involved and the particular controls used.
- (3) No extra storage or time is required to describe data which the user does not desire to protect.
- (4) The method is machine-independent and also independent of file structure. The efficiency of each implementation depends mainly on the adequacy of the formulary method for the particular data structures and application involved.
- (5) The discussion above illustrates the modularity of the formulary mode.

ACKNOWLEDGMENTS

This paper is a condensation of a Ph.D. dissertation at the Stanford University Computer Science Department. The author is deeply indebted to Professor William F. Miller for his encouragement and advice during the research and writing that went into it. Many other members of the Stanford Computer Science Department and the Stanford Linear Accelerator Center also contributed their ideas and help, in particular, John Levy, Robert Russell, Victor Lesser, Harold Stone, Edward Feigenbaum, and Jerome Feldman. The formulary idea was initially suggested by the use of syntax definitions ("field formularies") for input/output data descriptions as described by Castleman.³³

REFERENCES

- 1 P S CRISMAN (EDITOR)
The compatible time-sharing system—a programmer's guide MIT Press Cambridge Massachusetts 1965
- 2 J D BABCOCK
A brief description of privacy measures in the RUSH time-sharing system
Proc AFIPS SJCC Vol 30 pp 301-302 Thompson Book Co Washington D C 1967
- 3 B W LAMPSON
Dynamic protection structures
Proc AFIPS FJCC pp 27-38 1969
- 4 F J CORBATO V A VYSSOTSKY
Introduction and overview of the Multics system
Proc AFIPS SJCC pp 185-196 1965
- 5 L J HOFFMAN
Computers and privacy: a survey
Computing Surveys Vol 1 No 2 pp 85-103 1969
- 6 C ARVAS
Joint use of databanks
Report No 6 Statistiska Centralbyran Stockholms Universitet Ukas P5 Sweden 1968

- 7 H W BINGHAM
Security techniques for EDP of multilevel classified information
Document RADC-TR-65-415 Rome Air Development Center Griffiss Air Force Base New York 1965
- 8 C WEISSMAN
Security controls in the ADEPT-50 time-sharing system
Proc AFIPS FJCC pp 119-133 1969
- 9 D K HSIAO
A file system for a problem solving facility
Ph D Dissertation in Electrical Engineering University of Pennsylvania Philadelphia Pennsylvania 1968
- 10 M G STONE
TERPS-file independent enquiries
Computer Bulletin Vol 11 No 4 pp 286-289 1968
- 11 R M GRAHAM
Protection in an information processing utility
Communications of the ACM Vol 11 No 5 pp 365-369 1968
- 12 J B DENNIS E C VAN HORN
Programming semantics for multi-programmed computation
Communications of the ACM Vol 9 No 3 pp 143-155 1966
- 13 J K ILIFFE
Basic machine principles
MacDonald and Co London England 1968
- 14 D C EVANS J Y LE CLERC
Address mapping and control of access in an interactive computer
Proc AFIPS SJCC Vol 30 pp 23-30 Thompson Book Co Washington D C 1967
- 15 J A FELDMAN
Aspects of associative processing
Technical Note 1965-13 Lincoln Laboratory MIT Cambridge Massachusetts 1965
- 16 R G EWING P M DAVIES
An associative processor
Proc AFIPS FJCC 1964
- 17 R G GALL
A hardware-integrated GPC/search memory
Proc AFIPS FJCC 1964
- 18 J MC ATEER et al
Associative memory system implementation and characteristics
Proc AFIPS FJCC 1964
- 19 J I RAFFEL T S CROWTHER
A proposal for an associative memory using magnetic films
IEEE Trans on Electronic Computers Vol EC-13 No 5 1964
- 20 V R LESSER
A multi-level computer organization designed to separate data-accessing from the computation
Technical Report CS90 Computer Science Department Stanford University Stanford California 1968
- 21 P D ROVNER J A FELDMAN
The Leap language and data structure
Proc IFIP 1968 C73-C77
- 22 T D FRIEDMAN
The authorization problem in shared files
IBM Systems Journal Vol 9 No 4 1970
- 23 C E SHANNON
Communication theory of secrecy systems
Bell System Technical Journal Vol 28 pp 656-715 1949
- 24 D KAHN
The codebreakers
Macmillan New York New York 1967
- 25 R O SKATRUD
The application of cryptographic techniques to data processing
Proc AFIPS FJCC pp 111-117 1969
- 26 W F MILLER L J HOFFMAN
Getting a personal dossier from a statistical data bank
Datamation pp 74-75 May 1970
- 27 E W DIJKSTRA
Cooperating sequential processes
Department of Mathematics Technological University Eindhoven the Netherlands 1965
- 28 A SHOSHANI A J BERNSTEIN
Synchronization in a parallel accessed data base
Communications of the ACM Vol 12 No 11 pp 604-607 1969
- 29 R S JONES
DATA FILE TWO—A data storage and retrieval system
Proc SJCC pp 171-181 1968
- 30 R H GIERING
Information processing and the data spectrum
Technical note DTN-68-2 Data Corporation Arlington Virginia 1967
- 31 L J HOFFMAN
The formulary model for access control and privacy in computer systems
Report 117 Stanford Linear Accelerator Center Stanford California 1970
- 32 A N HABERMANN
Prevention of system deadlocks
Communications of the ACM Vol 12 No 7 p 373 1969
- 33 P A CASTLEMAN
User-defined syntax in a general information storage and retrieval system
In Information Retrieval The User's Viewpoint An Aid to Design International Information Inc 1967

Integrated municipal information systems: Benefits for cities—Requirements for vendors

by STEVEN E. GOTTLIEB

BASYS, Inc.
Wichita Falls, Texas

INTRODUCTION

The Fall Joint Computer Conference's call for papers this year says that, "The scope of the conference will encompass the entire information processing field." It goes on to say that the primary theme is "the use of computers to improve the quality of life."

So many of the new projects we as individuals or as nations undertake today, we purport to be an activity, which, if accomplished, will improve the quality of life. If I appear to be suggesting that this phrase is a little overworked, then, indeed, I have made my point. The problem, however, is not with the phrase itself, but rather with the broad, all encompassing meaning the user frequently wishes to imply when he cannot ascribe specific benefits to the project in which he is engaged.

I would, therefore, like to describe for you a few of the direct and indirect benefits which I see accruing from the USAC program and specifically from the Wichita Falls project to create an Integrated Municipal Information System. Before doing that, however, let me address the goal toward which we in Wichita Falls are striving.

THE DEFINITION OF AN IMIS UNDERSCORES THE GOAL OF THE WICHITA FALLS PROJECT

The USAC program is aimed at cities whose population is between 50,000 and 500,000. The Wichita Falls project is one of only two aimed at the development of what is called a total Integrated Municipal Information System (IMIS).

Total—In this context, means that the system considers all aspects of municipal activity and includes such diverse things as:

- The maintenance of criminal history information on previously convicted law breakers or of health

or welfare records for those persons receiving treatment or aid.

- The generation and posting of utility bills
- The posting of the general and subsidiary ledgers
- The maintenance of land use records
- The development and maintenance of the voter role

Integrated—Refers to the development of a unified, multi-functional data base; that is, a data base shared by all the generators and authorized users of data within the municipal government. Frequently in cities, as in any large complex organization, there is a multiplicity of requirements for the same data. Too frequently, however, these requirements are satisfied by each user independently collecting and storing the data for himself. The tax assessor, the fire department, and the building inspector, for example, all require similar information about buildings including such things as:

- Address
- Dimensions
- Construction type
- Number of access ways, etc.

There appears to be no reason why this data cannot be collected by only one of these departments on a single inspection and made available to the others. Integrated, in the context of this system, then really includes not only the development of a unified, multi-functional data base, but also the development of unified multi-purpose data collection and dissemination methods.

Municipal—Implies that the system concerns itself only with what is carried on by the city government. In this case, however, that's not an adequate definition. The city, though in many ways seemingly autonomous, has many interrelationships with other institutions. A city, for example, must interact with and be responsive to the needs of both the county and state in which it is

located. Additionally, cities must also interface with outside organizations or special districts including:

- Independent school boards
- Water districts
- Citizen or civic organizations
- Councils of Governments
- Economic development districts

In addition to these and others, cities must also be responsive to the many reporting demands placed on them by the Federal Government. Municipal is meant, therefore, to include not only the organizational units internal to the city government but all organizations with which the municipality must interface.

Information—Again, the scope of the word is broad and refers not only to the traditional information requirements of top management but perhaps more importantly to the information requirements of both middle management and those who carry out the day to day activities in which a city is engaged.

System—This includes the aggregation into a single functioning mechanism of not only all the pieces to which I have referred but also one other key element. That is, most cities (in the 50-500,000 population range), unlike many other large complex organizations, do not really have a large number of single process (payroll and billing type) applications suitable for computerization. Those cities rather have many multiple process applications. There are many departments which need and process considerable information; that is, they act on a large number of information categories but they generally do so relatively infrequently. In Wichita Falls, a city of only 100,000, we have, for example, identified well over 6000 individual data elements which are used in multiple combinations and must be kept readily accessible. We have not, however, been able to identify any single transaction type beyond conventional utility collection which could be considered to have a high transaction rate. This imposes a few unique design considerations in that no one application's file requirements clearly dominate the data base design.

To use their information effectively, cities frequently have to have a large number of people who "massage" the information, putting it in a form useful for managerial decisions ranging from "what are my budget requirements for next year" to "which of the traffic signals should have preventive maintenance performed." The system we are building includes the first step toward the solution of this problem by making more effective use of the city's valuable personnel resources. This will be accomplished by allowing the computer to make some, and perhaps many, of the routine decisions which too frequently occupy so much

of a manager's time. By way of example of such decisions, consider the time spent in determining:

- Which properties in the city should be reappraised
- Which vehicles and equipment now require preventive maintenance
- What is the best schedule for preventive maintenance considering skill requirements and available resources
- Which persons should be sent notifications of their failure to pay tickets or their need to come in for an additional medical examination

With this perspective in mind, the goal of the Wichita Falls IMIS can be stated as the design, development, and implementation of a multi-functional data acquisition and storage system which is capable of making routine decisions required during the daily operation of the municipality. Toward this end, we in Wichita Falls have made considerable progress.

TWO PHASES OF THE WICHITA FALLS PROJECT HAVE BEEN COMPLETED

At this time our project is one-third over and we have completed both the Analysis and Conceptualization Phases and are well into the Design Phase. As part of an early demonstration to prove the feasibility of building and using an integrated data base in a municipal environment, we will by November 1971 have implemented both an automated purchase order and vendor performance application, as well as an automated tax assessment update application.

TRANSFERABILITY IS A KEY ASPECT OF THE USAC PROGRAM

Transferability is the principal justification for the Federal Government's funding the IMIS development. Transferability may be simply defined as the expectation that the system, or at least significant parts of it, will be usable by other cities.

Having progressed to the point we are today, I can say with assurance that there are worthwhile products already derived from the Wichita Falls project which in fact are transferable and could be used by other cities. These products can be viewed as belonging to one of three categories:

- *Concepts*—The new ideas which have been developed through the project
- *Methodology*—The techniques which enabled us to develop our programs or concepts

- *Programs*—The actual computer programs that result from the Development Phase

A point to be made, however, is that the closer we get (on the above scale of products) to computer programs, the more difficult or lower the probability of direct transfer to another city, while the closer to concepts, the higher the transferability is expected to be.

With the recognition, therefore, that one of the key issues in this project is the development of products which are transferable, let me discuss the relationship to other cities of two major products which have already been developed:

- The Analysis Phase documentation
- The Conceptualization methodology

The documentation of the Analysis Phase of the Wichita Falls project, which is over 6500 pages and required some ten man-years to compile and produce, for the first time provided a comprehensive view of all the activities which go into making a "real" city work. It provides, to prospective city managers and other students of public administration, an additional perspective on the complexity of a successfully operating municipal government.

Additionally, the documentation provided Department Heads and Division Directors in the City of Wichita Falls an opportunity to more fully examine the operations of their organizational unit. Further, it served as a "before" snapshot of municipal operations to be compared with an "after" shot from the resulting design, development, and implementation documentation. This comparison will enable costs to be compared between some of the old and new systems. The final, but by no means least significant, benefit to be derived from the analysis is the fact that there is now a document available which another city can use as a basis on which it can perform a far less costly analysis of its own activities. Specifically, another city need only prepare exception documentation for those activities which it provides and which are significantly different from those in Wichita Falls.

The methodology we developed in the Conceptualization Phase is referred to as the Top-Down Bottom-Up Approach. This methodology enabled us to formulate a general system design which, from what we have been able to determine so far, should be transferable to a large number of other cities.

The approach was as follows. Consider that the purpose of a city, simply stated, is to provide public services desired or demanded by its citizens or society. These services or functions, of which there are many,

include such things as:

- Protecting the people from those who break the law
- Providing water and sanitary services
- Providing for the transportation of people and goods

The total of these functions can be broadly grouped into four sectors (originally defined by USAC as sub-systems):

- Public Safety
- Human Resources Development
- Public Finance
- Physical and Economic Development

The functions themselves can be divided into components, and the components into applications in a typical hierarchical fashion similar to the organization of most municipal governments. This Top-Down hierarchy provides the overall functional framework which is to be served by the information system. To structure the information system, however, one must also look at the information requirements of each of the lowest level applications, including such things as:

- Maintaining traffic signals
- Determining if a given vehicle or person is stolen or wanted
- Putting out fires
- Processing platting changes
- Issuing parade licenses

If one then considers those applications which have common information characteristics one can begin to see the data exchange necessary for effective data sharing.

In the Wichita Falls project we aggregated those operational applications with what appeared to be the highest number of common information characteristics into what we called a DISC, a Decision Information Set Center. A DISC is a hypothetical module containing many processes, all related to the same or similar data. The DISC provides a convenient means of simultaneously considering the data and logical file requirements of many informationally related applications (Figure I). It was found that DISCs could be defined in three levels:

- Operational
- Operational/Analytical
- Analytical

The lowest level or operational DISCs provide the

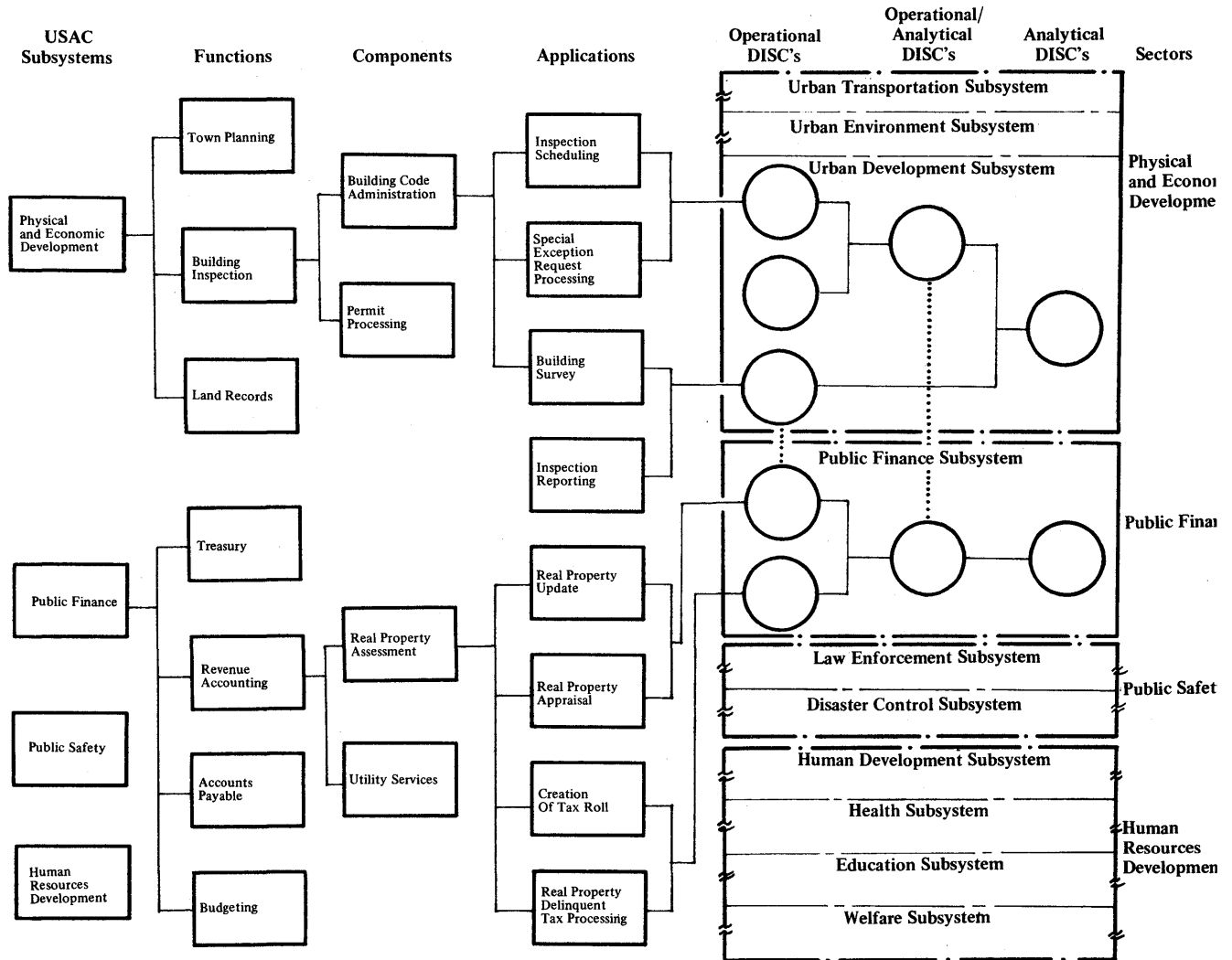


Figure 1

input to more analytical applications which could be aggregated into the higher level operational/analytical DISCs. Further, the output of these DISCs could in turn be aggregated into DISCs of exclusively analytical applications such as those associated with annual resource allocations or comprehensive planning. It was further found that the output of these analytical DISCs provided feedback to the first level operational DISCs, thus establishing essentially closed systems. These closed systems are, in effect, subsystems of the total IMIS. For the IMIS postulated in Wichita Falls, ten such subsystems were identified:

- Public Finance
- Disaster Control
- Law Enforcement

- Urban Development
- Urban Transportation
- Urban Environment
- Welfare
- Education
- Health
- Human Development

The significance of the subsystem is NOT that there are ten, since a different aggregation of the operational applications could change the number of subsystems slightly, but rather that a city can be viewed, informationally, as being comprised of a number of subsystems with associated data flows.

From the viewpoint of transferability, the Wichita Falls project's Conceptualization Phase resulted in

what we feel to have been a success. We have a concept and a methodology which are clearly transferable, but we also have a product, a general systems design which is also felt to be transferable, though clearly, modification will be necessary to accommodate, among other things, the variation in services provided by other cities.

THE DESIGN PHASE HAS PROVIDED NEW AREAS OF STUDY FOR SOFTWARE VENDORS

The Design Phase in which we are now heavily involved is beginning to provide some interesting problems for further study. Because we have not yet progressed far enough in the phase to discuss its products, I have chosen to mention a basic design philosophy and comment on some areas in which I believe the computer industry should provide some additional guidance.

Because of the complexity of the system to be built, and the desire to minimize data redundancy, while at the same time providing the data to multiple users, we chose to develop an integrated data base. In order to assist in the management of this data base, we further chose to implement a vendor supplied data base management system. The efforts of our project staff were then divided into three main areas:

- Data Base Design
- Application Design
- Application Programming (this is actually part of the Development Phase)

Clearly, neither of these areas can be considered in vacuum, nor do we really split the staff into three distinct groups. For the purpose of our discussion, however, it is appropriate to consider the three groups as operating independently but toward a common objective.

Toward that end then, the application design teams provide programming specifications and detailed flow charts to the programmer group while concurrently

providing data requirements to the data base group which designs the base by establishing the files and associated linkages. This over-simplification thus enables me to separate out and address only the data base portion of the Design Phase.

Initially, when we began the phase, I was surprised to find how little was generally known about data base management. Obviously, part of the problem lies in the fact that it is a relatively new field in which few people have had any experience.

What is particularly unfortunate, however, is not how little is known about data base management, but rather how little is known and how little effort goes into understanding the functions of cities (a prime sales target for computer and software vendors). All too often I was told how a given system could handle any data base problem the City might have. The substantiation was based on the fact that the system had just been implemented or was to be implemented by Company XYZ which, I was told, was bigger than Wichita Falls and had bigger files to work with. I have no doubt that most existing data base and file management systems are capable of handling relatively large files. I have considerable doubt, however, about their ability to efficiently process multiple, extensively linked files with individually low transaction rates and with the high number of different processes found in cities. What we have found thus far in our project (and the results are available to each of you) is that USAC-sized cities do in fact have unique data processing problems. It does not appear that these problems are necessarily more complex than those associated with industry. They are merely different.

I suggest to you, therefore, that you have in cities a major and virtually untapped market place, one which you can and should serve efficiently by studying their needs and solving their problems. Do this not by looking at cities as non-profit companies whose needs can be served by the old products and methods developed in the past, but because of their importance and because, in fact, their problems are different, by a fresh examination and by the development of products designed to meet their unique needs.

Geocoding techniques developed by the census use study

by CABY C. SMITH and MARVIN S. WHITE, JR.

U.S. Census Bureau
Los Angeles, California

HISTORICAL PERSPECTIVE

The Census Use Study (CUS) was established in September 1966 in New Haven, Connecticut. The emphasis of the study is on small area data, i.e., data relevant to areas smaller than a city. The CUS took the lead in geocoding by developing the DIME file and ADMATCH, an address matching system. The CUS at New Haven also developed the Health Information System to provide health planners with powerful statistical tools. In July, 1969, the Southern California Regional Information Study (SCRIS) was established in the Los Angeles area in order to transfer the experience gained at New Haven to a large urban area. SCRIS is continuing the geocoding work begun at New Haven and has embarked into other fields, while continuing to improve existing computer programs. For instance, SCRIS has produced an IBM 360/OS and an RCA SPECTRA 70 version of ADMATCH. A computer mapping system was also developed at SCRIS from the basic research activities carried on at New Haven. Investigations have begun on several other fronts including an extension of a fallout shelter study, the Summary Tape Retrieval Information Processor (STRIP), a generalized file matching system, and several special purpose tools related to small area data and geocoding. In addition, the New Haven CUS and SCRIS have produced a series of publications on these and other related topics for public information. SCRIS is producing a series of transportation related publications, which should be of interest to most transportation planners.

GEOCODING^{1,2}

By geocoding we mean the process of attaching relevant geographic codes to data which has some less useful geographic codes. Usually this means converting street address to some area code, e.g., census tract

through ADMATCH, but may involve more complicated tabulations, such as from street intersections to police precinct.

A wealth of data, useful for planners, is scattered in existing files, e.g., Assessor's files, welfare files, and motor vehicle registrations. Accessing the data is often difficult or impossible due to lack of useful geographic codes, confidentiality rules, etc. One can often surmount these barriers through geocoding, usually ADMATCHing the data file to determine census tract and block from street address.

An agency responsible for confidential information about individuals is frequently willing to release summaries by some large enough geographic area. The Bureau of the Census is a good example of such an agency. Census data on individuals is confidential and suppressed, but tabulations by block group, census tract and other geographic areas are released. The SCRIS staff has successfully geocoded a number of such files including welfare and building permit files.

Often an agency will release tabulations of data by a geographic area important to that agency but not to other users. For example, welfare departments may release tabulations by welfare district, which probably do not conform to other statistical or planning zones. Worse, welfare district boundaries may change rapidly as caseloads vary and thus destroy historical continuity and render analysis by other zones prohibitively expensive. Provided street address is available, geocoding easily solves these problems.

ADMATCH^{1,3}

ADMATCH was designed to perform this type of geocoding, obtaining some geocode like census tract from street address. ADMATCH operates by linking two logically connected files, a data file and a reference file (see Figure 1). The data file contains a street address (or range of addresses) and the interesting

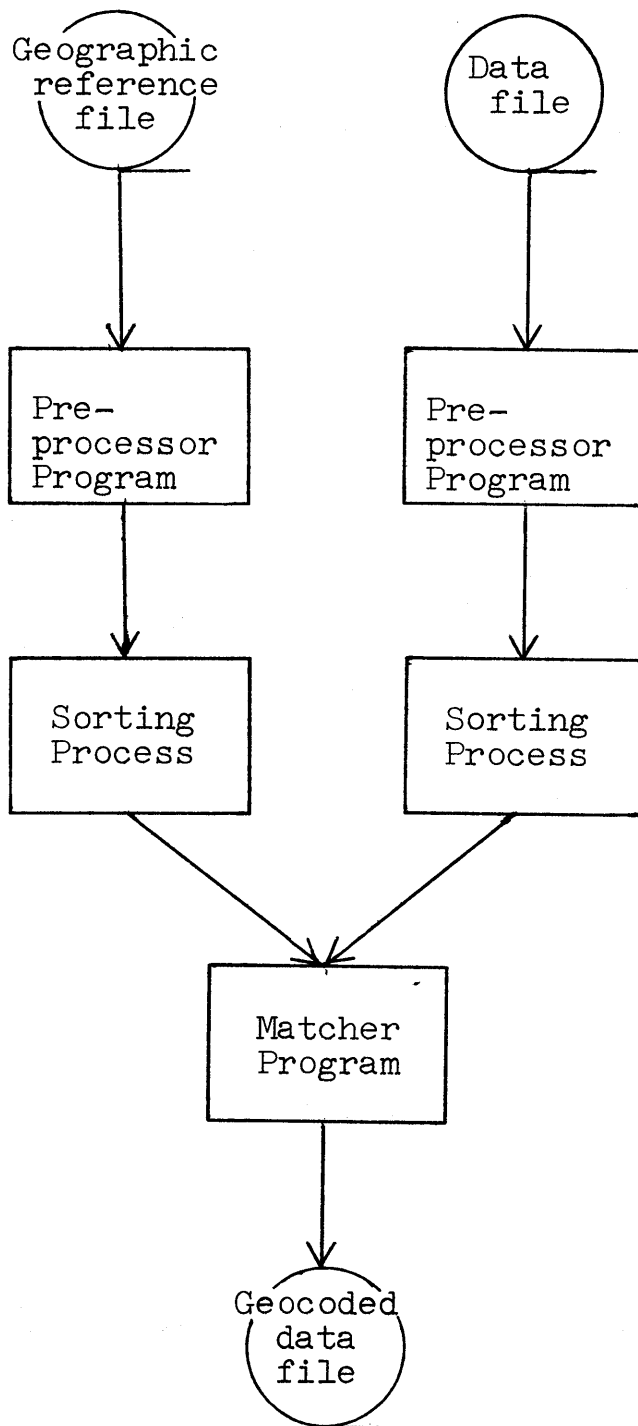


Figure 1—ADMATCH system overview

data; the reference file, a geographic base file, contains street address and the corresponding geocodes. ADMATCH has rendered a great deal of otherwise inaccessible data easily accessible.

ADMATCH performs this file linkage in two phases. The Preprocessor analyzes a character string address

according to syntax and keywords specified by the user and creates a standardized version of the address called the "match key." The Matcher compares the data record match key to all the reference file records with the same street name and selects the best match. The best match is determined according to a weighting scheme determined by the user. Reliable performance and cost estimates for the IBM 360/DOS ADMATCH are available from CUS Report No. 14, *Geocoding with ADMATCH: A Los Angeles Experience*. The OS ADMATCH is much more efficient than the DOS version, but specific cost benchmarks are not yet available.

GEOGRAPHIC BASE FILES

The reference file required by ADMATCH is one of a class of files called Geographic Base Files (GBF). Since urban planning is so extensively related to geography, close attention should be paid to GBFs and their multitude of applications. A GBF is minimally an extended correspondence table for two or more geographic codes. However, a GBF may be much more complicated, viz., it may reflect any geography related structure or information. For example, street network and land use information may be contained in a sufficiently finely resolved GBF.

There are a multitude of applications for GBFs. A very important application, providing linkage between otherwise incommensurate data, was already discussed. A GBF can serve as a geographic base for information systems in a number of ways. First, the desired information (e.g., street or area maintenance information) can be coded directly into the GBF. Or a GBF could act as an index to another file or a series of files, i.e., the GBF might contain pointers to other files. The Address Coding Guide (ACG) is a GBF in wide-spread use. Each record in the ACG represents one block face and contains address range information and geocodes like census tract, county and place.

The ACG was developed by the Bureau of the Census in part to facilitate the mail-out mail-in census but more importantly for us to provide a GBF with nation-wide standards. The Metropolitan Map Series (MMS), which serves as a source for ACG coding, was the first step in standardization on a nation-wide basis. Metro Maps are produced by the Census Bureau for each Standard Metropolitan Statistical Area (SMSA) in the United States.

DIME^{4,2,5}

The Dual Independent Map Encoding (DIME) file is the most complete type of GBF developed to date.

The DIME concept was developed and first implemented at the New Haven CUS. Each record in the DIME file represents a street *segment* bounded by a node at each end. Nodes are placed on a map (Metropolitan Map Series) at each interesting place, i.e., each street intersection, each intersection of a street with an important nonstreet feature like railroad crossings, and curves. Furthermore, the DIME file contains the coordinates of each node. The node and coordinate information renders the DIME file extraordinarily useful. The DIME file has all the applications of any GBF and a great deal more than most. In fact, as far as network related geographic features are concerned, the DIME file has the ultimate form for a complete GBF. It can contain all relevant information, provided it is finely enough resolved. For example, the DIME file contains geocodes for both left and right sides of the segment, such as block number, and may also contain street usage codes.

Some of the most interesting applications for the DIME file are transportation related. Transportation planners routinely perform network and node analyses on traffic flow. The DIME file in conjunction with DAM (DIME Aggregation Manager), a system to abstract higher level networks, can be used to analyze traffic networks at any desired level of detail. SCRIS is presently investigating the possibility of producing from the DIME file, a network file for input directly to the Bureau of Public Roads Urban Transportation Planning system. Other transportation related applications of the DIME file are automated routing of busses, etc., either real time or batch, traffic flow modeling, and intersection or node related studies. Nearly all police and traffic safety records are coded to intersection, like Hollywood and Vine. An intersection file can be constructed from the DIME file.

Since the DIME file contains coordinates for each node, a number of other applications are possible. Using the DIME file as a base for computer mapping is one of the more exciting applications. In fact, the CUS staff at both New Haven and SCRIS have mapped DIME files themselves for editing purposes. More will be said below about computer mapping. Areas and centroids can easily be calculated for areas bounded by streets, using the DIME file. The CUS is studying the feasibility, in terms of cost, of further DIME applications.

Two additional features of the DIME file should be noted. The DIME file offers special advantages for editing and updating. These boil down to the fact that the DIME file is an interconnected network of records and that these records are unequivocally tied down to specific lines on a map.

Since DIME records are tied to maps, records may

be referenced with only the map as a source. Thus, the clerical step of finding a serial number or other key in a listing is eliminated. A significant source of errors and cost is also eliminated. In the case of an ACG file, for instance, a clerk must resolve ambiguities that arise when the same street name occurs on two faces of the same block by scanning a listing.

The fact that DIME records form a connected link network means that topological edits may be performed. For example, DIME records may be chained around a block or census tract. Boundaries that do not close are flagged as errors and corrected. Since nodes occur at every intersection, no two segments represented by DIME records should intersect. Whether two segments intersect is easily determined by a vector cross product calculation. Thus, nodes with incorrect coordinates can be located by noting that segments containing them intersect with others.

COMPUTERIZED RESOURCE ALLOCATION MODEL (CRAM)^{5,6}

We have seen a number of applications for GBFs in general and the DIME file in particular. The most sophisticated application being developed by the CUS is the Computerized Resource Allocation Model (CRAM). Basically, CRAM is a generalized system for determining the service areas for a set of facility locations. In its most general form, service areas are constrained by facility capacities and travel times. CRAM is a refinement and extension of the NAPS (Network Allocation of Population to Shelter) system developed by System Development Corporation in connection with a CUS contract with the Office of Civil Defense.

The system uses a DIME file as its network base and in addition requires demand and facility capacity inputs. The problems which can be attacked through CRAM run from simple fixed source districting problems such as school district boundaries, park planning, and site location to much more complicated problems of emergency vehicle routing, delivery or bus route planning and freeway location studies.

CRAM uses a modified version of the Moore Shortest Path Algorithm to do its geographic analysis. The Moore technique (not CRAM's version) finds shortest paths between points using a point-to-point incidence matrix (with associated distances or costs) as its road map. In this technique, the network is viewed as a set of interconnected nodes, the connections being links. The DIME file, however, directly represents a set of interconnected links, the connections being nodes. The DIME file can be modified to fit the Moore tech-

nique easily enough, but there are advantages to modifying the technique to fit the DIME structure.

The advantage in modifying the technique arises mainly in the allocation of demand. Demand on a facility is generally the number of persons desiring to use the facility. Information about numbers of persons usually relates to blocks or block groups—not nodes or links. However, the disaggregation of persons to links is much more reliable than disaggregation to nodes. This is true because links have length and the disaggregation may be varied according to length, but not for nodes.

UNIMATCH⁷

The applications for GBFs and file linkage in general are so extensive that SCRIS has begun the development of a generalized file linkage system—UNIMATCH. UNIMATCH is structurally similar to ADMATCH in that it consists of a STANDARDIZER and a MATCHER. However, UNIMATCH will not be limited to matching street address. Instead, street intersections, major traffic generators or any logical connection may serve to link two files. This generality is achieved by allowing the user to specify what fields to compare, what comparisons to make (e.g., character, numeric or parity comparisons), what significance to attach to a success or failure to compare and finally what action to take depending on the level of success of the comparison. That action might be further comparisons or the copying of selected fields from one file to another. The impetus for designing and implementing UNIMATCH comes from transportation planning needs. Originally, the system was to be named TRAM (Transportation Related Address Matcher). Great efforts were made to name the system TROLLEY, but no suitable words to fit the acronym made themselves obvious.

COMPUTER MAPPING^{8,9}

The CUS has investigated a variety of available computer mapping programs and has developed a mapping system—GRIDS (Grid Related Information Display System) at SCRIS to meet needs that were not being met. The existing computer mapping systems required quite a bit of programming ability, a good deal of data preparation, and in many cases required large and expensive computing facilities. For these reasons, mapping of data was left mainly in the hands of draftsmen. For these same reasons, GRIDS, which is a system for producing inexpensive printer maps quickly and easily, was created.

GRIDS is an especially good analytical tool for planners. Voluminous data is much more easily assimilated in map form and GRIDS handles finely resolved data especially well. Also, each map costs only \$5 to \$10. Furthermore, GRIDS will run on nearly any system with FORTRAN compiler—it has run on an IBM 360 model 30 with 32K bytes of storage.

The finest unit of resolution for GRIDS is one grid cell. The system produces maps by dividing the area to be mapped into a network of rectangular grid cells. A grid cell may be as small as one printed character or as large as 55×55 characters.

GRIDS is very flexible both in digesting input data and producing maps. There are three types of maps available: (1) shaded, in which data values are represented by overprinted characters of varying darkness; (2) density, in which a character is splattered randomly throughout each grid cell with more characters representing higher values; (3) value, in which the actual data value is printed in each cell. Figure 2 is an example of a GRIDS shaded map.

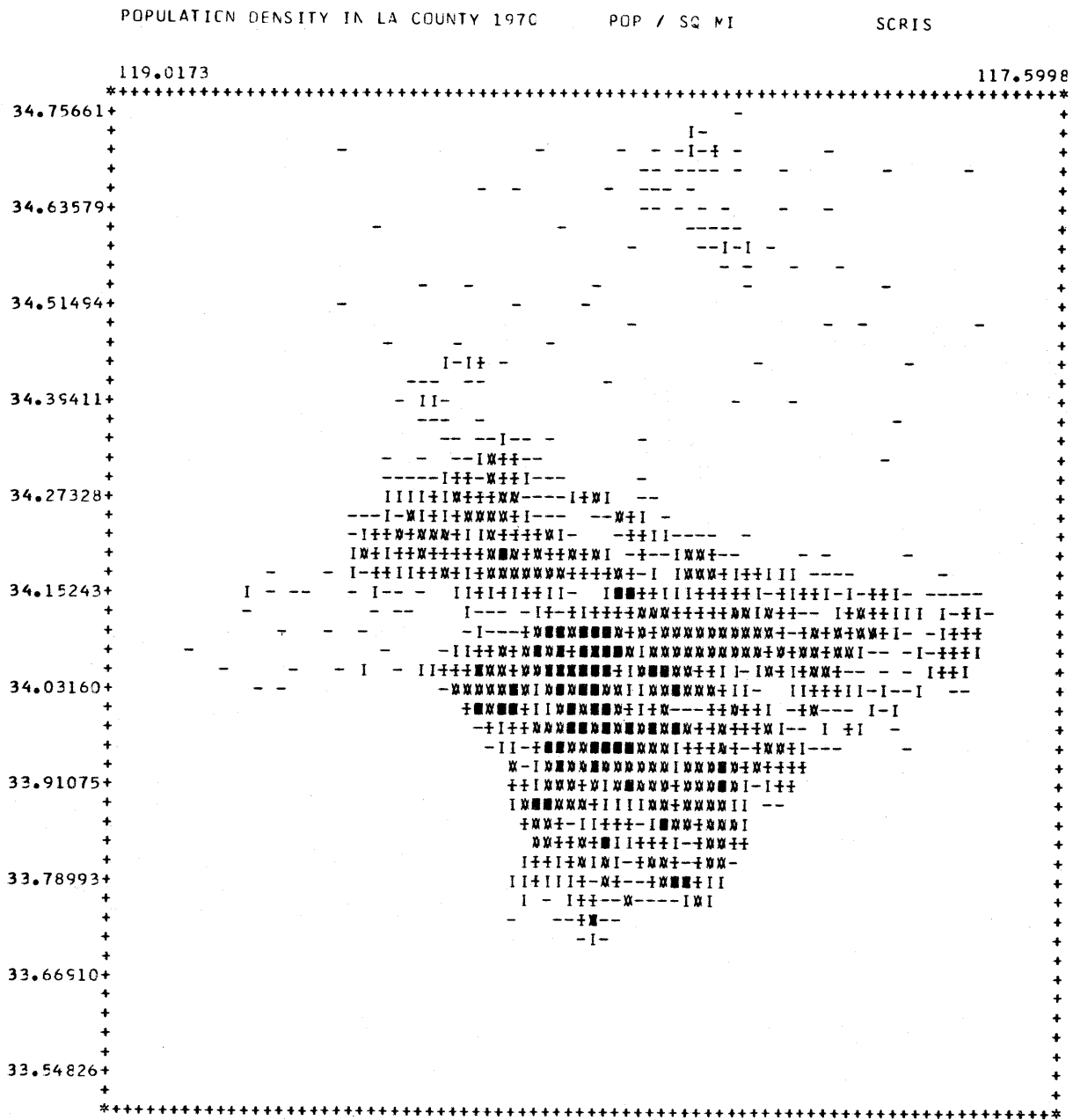
GRIDS accepts as many as 8 input variables and 2 coordinates to be mapped, provides for manipulation of the data in any desired way, and produces up to five different maps for each run. No previous preparation of the data is necessary and no knowledge of the data values is necessary. GRIDS provides for data manipulation through MAPTRAN, a built in FORTRAN-like programming language, or a user exit routine if necessary.

GRIDS is especially easy to use because all specifications are free format keyword type and there are many default values for the user with simple needs.

GRIDS has many important applications. Because GRIDS is inexpensive and flexible, it is a very good analytical tool. GRIDS may also be used as a prelude to more cosmetic and thus more expensive computer mapping techniques.

The CUS has also used, and is beginning a further study of, the Geospace Plotter. This plotter is a Cathode Ray Tube Photographic device with very good resolution and a selection of 32 intensity levels. One may select either 100 or 200 dots per inch resolution, with each dot addressable. Geospace plotter maps approach the quality of maps produced by a draftsman. These maps are excellent for public display and published reports, e.g., to decision makers at all levels.

The software supplied by the Geospace Corporation called ALPACA is very similar to software for pen plotters. SCRIS is presently improving the efficiency of ALPACA, particularly for mapping applications. SCRIS is also developing a mapping system, which will be particularly useful with the DIME file. Figure 3, a Geospace Plotter map of New Haven using the DIME



--- GRIDS --- (GRID RELATED INFORMATION DISPLAY SYSTEM) --- GRIDS ---

POPULATION DENSITY FOR LOS ANGELES COUNTY 1970
FROM 1970 MEDLIST

MSW 6/71
SCRIS

POPULATION / SQ MILE

I*****I
<---10MI--->
MINIMUM CELL VALUE(S) 0.0
MAXIMUM CELL VALUE(S) 0.222065E 05

DEGREES LATITUDE NORTH
DEGREES LONGITUDE WEST

LEGEND:

-----	IIIIIIII	+++++	XXXXXXXX	■■■■■■■■
-----	IIIIIIII	+++++	XXXXXXXX	■■■■■■■■
-----	IIIIIIII	+++++	XXXXXXXX	■■■■■■■■
-----	IIIIIIII	+++++	XXXXXXXX	■■■■■■■■
-----	IIIIIIII	+++++	XXXXXXXX	■■■■■■■■

.000000	1999.999	3999.998	6999.997	11999.99
1999.999	3999.998	6999.997	11999.99	22206.50

FREQUENCY	261	151	192	192	58
-----------	-----	-----	-----	-----	----

Figure 2—Sample GRIDS output using the shaded option. The legend displays each shade used, the corresponding data value range and frequency (count of grid cells) for each shade

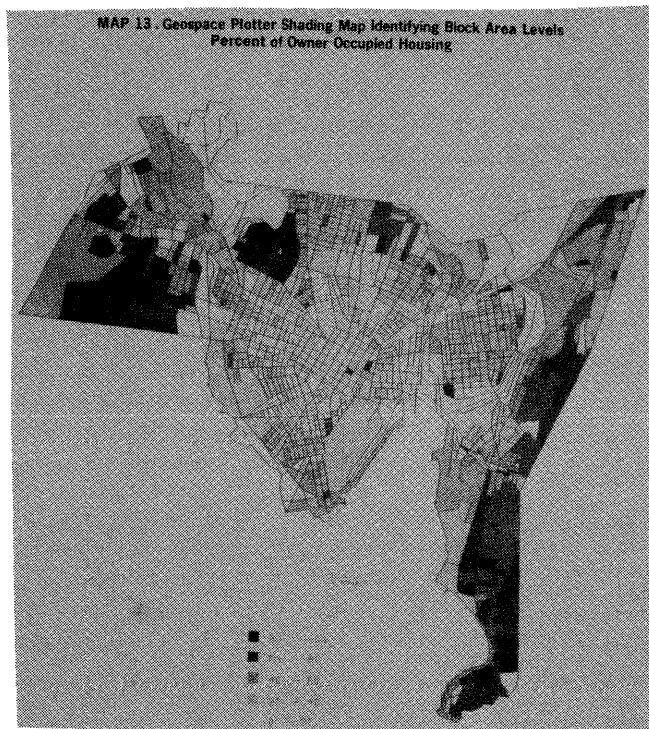


Figure 3—Geospace plotter shading map (Taken from Census Use Study Report No. 2, p. 41)

file for the street network base, is a sample of the quality obtainable through these techniques.

HEALTH INFORMATION SYSTEM^{10,11}

The Health Information System (HIS) was developed in New Haven by the CUS. Many of the computer tools mentioned above are incorporated into the HIS. Initially, the HIS concentrated on maternal and child care but has now become a more general information system for many health related fields, such as social pathology and health care delivery systems. The purpose of the system is to pinpoint geographically those neighborhoods where there is a significant health risk and define the characteristics of the population to provide health planners with analytic tools for approaching the solution of health problems. The HIS gathers data from a variety of sources and analyzes that data through advanced statistical techniques. Naturally, HIS techniques may be transferred to other fields such as education and crime.

Briefly, the analysis proceeds as follows. Nearly 300

data items are collected from such sources as the First Count Summary Tapes, Vital Records, Mental Health files, and the Mental Retardation Register. These data are linked through ADMATCH and summarized by block group. Correlational, factor and further multi-variate analyses are performed on the data to produce a few constructs or typologies. A topology is a synthesis of many items, ordered logically by their contribution to the whole. The resulting typologies themselves and maps displaying which block groups are typical for a given typology (i.e., which block groups rank in the top quartile among all the block groups with respect to this typology) are extremely useful to planners.

The CUS plans to also perform cluster analyses on the 1970 data base to obtain a more homogeneous grouping of neighborhoods. A time-series analysis on the 1967-1970 data will also be carried out. The emphasis will be on comparing changes in configurations of data items rather than just one dimensional comparisons. For example, illegitimacy is correlated with a number of other variables including family disorganization, high welfare roles, poor health of mother and child, and low socio-economic status. This time-series analysis will consider not only illegitimacy but the configuration of variables associated statistically with illegitimacy.

The HIS is now being expanded to Los Angeles through SCRIS and UCLA. The HIS methodology is already being used in Nebraska and Iowa by the Comprehensive Health Planning Council there. The HIS is partially implemented in a dozen places around the U.S.

PRESENT ACTIVITIES¹²

SCRIS is presently engaged in a wide array of special purpose activities to demonstrate many data processing capabilities to the planning community and to investigate the effectiveness-cost relationship of these activities. Two of these, the Summary Tape Retrieval Information Processor (STRIP) and SCRIS Report No. 5, will serve to indicate the nature of the investigations.

STRIP consists of several related programs to select Census Summary Tape records by geographic code and further select certain data items and produce tabulations of these items. STRIP operates in two phases. The first phase performs the selections and reformats the data to binary for more efficient processing later. Naturally, these intermediate data sets are available to the user for his own processing. The second phase produces set reports as specified by the user.

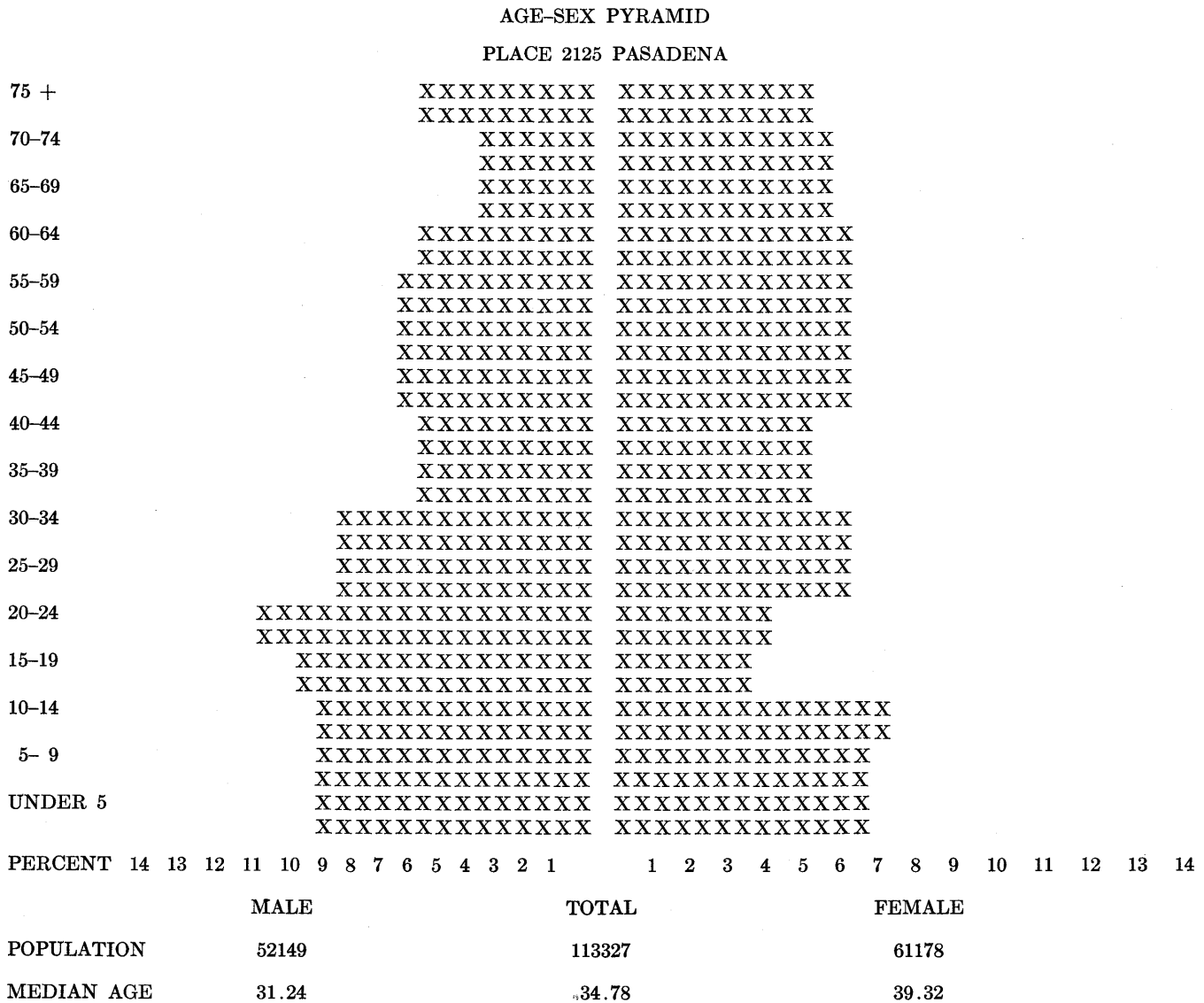


Figure 4—Age-sex pyramid for Pasadena, California (Taken from SCRIS Report No. 5)

SCRIS Report No. 5, *1970 Census Data: Characteristics of Cities and Unincorporated Places*, is a demonstration of how useful census data can be accessed and displayed. The programs used to produce the report are being packaged for distribution. The report contains tabulations of certain demographic characteristics by place. These include white population, Negro population, rents and home values. Age-sex pyramids were also tabulated for each place. An age-sex pyramid is a two-way graph, percent male increasing to the left from the middle, percent female increasing to the right and age increasing vertically. Age-sex

pyramids are used extensively by planners as they indicate information on the age and flavor of a community.

A careful examination of Figures 4 and 5, a sample from SCRIS Report No. 5, reveals a great deal about Pasadena. Unfortunately, there is no information whether the ladies over 75 are little or not.

SUMMARY

The CUS is involved in a college of data processing activity all related to small area data and much of it

TOTAL POPULATION 113327				TOTAL DWELLING UNITS 47093			
DATA ITEM	COUNT	PERCENT	RECORDS SUPPRESSED	DATA ITEM	COUNT	PERCENT	RECORDS SUPPRESSED
WHITE POPULATION	90446	79.8	0	1-UNIT STRUCTURES	28431	60.4	0
NEGRO POPULATION	18256	16.1	0	2 OR MORE UNIT STRUCTURES	18578	39.5	0
INDIAN POPULATION	281	0.2	0	MOBILE HOMES	80	0.2	0
OTHER SPECIFIED RACES	3488	3.1	0	OVER CROWDED UNITS	2243	4.8	0
REPORTED OTHER RACE	856	0.8	0	UNITS LACKING PLUMBING FACILITIES	984	2.1	0
OWNER OCCUPIED DWELLING UNITS	19483	41.4	0	UNITS LACKING KITCHEN FACILITIES	1106	2.3	0
RENTER OCCUPIED DWELLING UNITS	25170	53.5	0	POPULATION IN OVERCROWDED UNITS LACKING PLUMBING FACILITIES	209	0.2	0
VACANT DWELLING UNITS	2436	5.2	0				
VALUE OF OWNER OCCUPIED UNITS				RENT OF RENTER OCCUPIED UNITS			
	COUNT	PERCENT			COUNT	PERCENT	
LESS THAN 5000	38	0.2		LESS THAN 40	321	1.3	
5000- 9999	299	1.7		40-59	1340	5.5	
10000-14999	1615	9.0		60-79	4490	18.5	
15000-19999	3322	18.6		80-99	5304	21.8	
20000-24999	3077	17.2		100-119	4103	16.9	
25000-34999	4417	24.7		120-149	4441	18.2	
35000-49999	2901	16.2		150-199	2512	10.3	
50000 +	2190	12.3		200-299	1258	5.2	
				300 +	566	2.3	
MEDIAN	26300			MEDIAN	103		
RECORDS SUPPRESSED	0			RECORDS SUPPRESSED	0		
TOTAL RECORDS			1				

Figure 5—Selected 1970 Census data tabulations for Pasadena, California (Taken from SCRIS Report No. 5)

connected to geocoding and geographic analysis. A great deal of work has been done to provide planners with tools needed to analyze local and census data. ADMATCH has unlocked much information; UNIMATCH promises to unlock considerably more. The DIME file is the basis not only for file linkage through ADMATCH but also for computer mapping and quite sophisticated analyses and modeling such as CRAM.

We conduct research at the CUS in close conjunction with planners and others actually using the data to make the results of our research immediately relevant to the needs of users. CUS research serves as a foundation for further research both by CUS and others.

REFERENCES

- 1 ———
ADMATCH users manual
US Bureau of the Census
Census Use Study Washington DC 1970
- 2 J P CURRY G FARNSWORTH
The DIME geocoding system
US Bureau of the Census
Census Use Study Report No 4 Washington DC 1970
- 3 M JARO
Geocoding with ADMATCH
US Bureau of the Census
Census Use Study Report No 14 Washington DC 1970
- 4 R CRELLIN G FARNSWORTH
ACG-DIME updating system: An interim report

-
- SCRIS Report No 4
Los Angeles California 1970
- 5 G FARNSWORTH
DIME applications and the computerized resource allocation model
American Statistical Association Joint Statistical Meetings Detroit Michigan December 29 1970
- 6 G FARNSWORTH
Computerized resource allocation model
Unpublished paper SCRIS 1970
- 7 M JARO
Conversation concerning UNIMATCH
- 8 _____
Computer mapping
US Bureau of the Census
- Census Use Study Report No 2 Washington DC 1970
- 9 M JARO
Grid related information display system: GRIDS
To be published by the US Bureau of the Census
- 10 J DESHAIES
Health information system
US Bureau of the Census
Census Use Study Report No 7 Washington DC 1969
- 11 J DESHAIES
Conversation on recent health information system developments
- 12 _____
1970 census data: Characteristics of cities and unincorporated places
SCRIS Report No 5 Los Angeles California 1971

URBAN COGO—A geographic-based land information system*

by BETSY SCHUMACKER

Massachusetts Institute of Technology
Cambridge, Massachusetts

INTRODUCTION

COGO, a geometric problem solving system, has been in existence for ten years in its basic form and for four years in its more advanced form. COGO provides a command-structured language and a set of processing routines to define and describe such geometric objects as points, curves, courses, chains, and vertical profiles, and to perform geometric computations such as locations and intersections to find new point coordinates. It also includes file capabilities to enable users to gradually derive problem solutions over a period of time as well as to provide the ability for different users to try different problem solutions against the same set of data. It provides dynamic memory management and dynamic program management as a subsystem of the ICES System.

URBAN COGO expands upon ICES COGO in several ways:

- New geometric objects can be defined, namely blocks, regions, networks, and three dimensional objects such as building and street overpasses and underpasses.
- Expansion of file capabilities and the provision for hierarchies of files and subdivisions of files.
- Graphical output capabilities including both soft and hard copy displaying of objects or groups of objects with or without translation, rotation, or magnification, density mapping, selective mapping, and detailed mapping with full annotations.
- Graphical input capabilities by digitizing on a display screen or digitizing from hard copy on a flat-bed plotter-digitizer.

- Data attribute capabilities to associate textual data (e.g., land use information) with any or all of the geometric objects.
- Processing facilities for the data attributes to be able to sort, tabulate, or report on such data, and groupings and analyses to perform statistical tests on such data.

It is the author's belief that the combination of the original COGO concepts and the expansions summarized above form the basis for URBAN COGO to provide the base and the direction for urban information systems of the future.

GEOMETRIC OBJECTS

The geometric objects currently allowed in the system are

points
curves
courses
chains (or parcels)
profiles
networks
blocks
regions

Points

Points are the basic geometric unit of the system and have absolute values associated with them. Points are identified by number and can have x, y, and z coordinates stored as their values.

* The work reported herein was supported in part by grant no. GK-25622X from the National Science Foundation.

STORE POINT 17 X 113.21 Y 7521.831 Z 37.512

Points can be identified and have values stored for them by use of the STORE command (above) or by use of any of the LOCATE or INTERSECT commands (below) or by digitizing them on a graphic display unit or on a plotter digitizer unit.

LOCATE 512 FROM 17 DISTANCE 153.91
BEARING N 37 23 E
INTERSECT CHAIN 'A' WITH NETWORK
'C' POINT 801

Curves

Curves are also absolute objects, are identified by number, and are either circular arcs, circles, or spiraled arcs with circular arcs. They are planimetric and are defined by storing or by digitizing.

Networks

Networks are relative objects and are identified by an 8-character name. Networks consist of links which, in turn, are defined by begin and end nodes which are point numbers. Links can be singly-directional or dual-directional and can vary from network to network. Networks can be defined by storing or digitizing.

Courses

Courses are relative objects defined by the point number at each end. They are identified by a four-character name and defined by storing or digitizing.

Chains

Chains are relative objects, identified by an 8-character name, and defined by storing or digitizing. Chains consist of points, curves, and courses, either with continuous boundaries or with gaps. While chain is the general name for this object, a more specific name for some applications is parcel, which name the system recognizes as a synonym for chain. The word parcel implies the same thing as a parcel of land which is the smallest legally recognized unit of land.

The points and curves which define a chain locate the chain in space; the sequence of the items which define it (i.e., the sequence of the points, curves, or courses) define the topology of the chain.

STORE CHAIN 'A' POI 159 CURVE 7 POI
153 CUR 91

Blocks

Blocks are relative objects identified by an 8-character name and defined by storing. Blocks consist of chains. One example of a block is a street block which consists of parcels (chains) of land.

STORE BLOCK 'D' CHA 'B' 'A' 'F' CHA 'G'

Regions

Regions are relative objects identified by an 8-character level and an 8-character name. Any number of levels of regions may be defined and stored. A first level region consists of blocks, a second level region consists of first level regions, etc. This capability permits the user to go from very fine geometric data to very gross geometric data (in terms of size) in any way he wishes.

Object grouping

Standard parcels of land may be stored as chains, street blocks containing these parcels may be stored as blocks, census tracts containing these blocks as first level regions, counties containing these census tracts as second level regions, states containing these counties as third level regions, the country as a fourth level region.

Users interested only in gross areas may start out with census tracts as chains, counties as blocks, states as first level regions.

In other words, by designing the system so that the user can be very accurate at the base level, any grossness of accuracy is also possible merely by defining the base levels to be something different. Thus, in the first use above point coordinates are stored as accurately as they can be measured by surveying and everything else has the same accuracy.

Thus, the accuracy achieved is not what the system imposes upon the user, but what the user himself imposes by his choice of base level and his own needs.

DATA ATTRIBUTES

Data attributes can be associated with any of the geometric objects. The allowable categories of attributes and range of values for these are defined as essentially a system setting for a particular execution of the system. The system setting aspect of this definition permits different groups of users to access the same

geometric data base with their own associated attribute data. Thus urban planners, transportation analysts, city managers, utility company inspectors, etc., can all use the same base geometric data of a city but utilize that attribute data which is meaningful and useful to him.

The allowable names for these attributes and the range of values each may have permits (1) the user to use nomenclature which is familiar to him and (2) the system to do editing "for free" on this data while it is being stored, updated, modified, or accessed.

Definition

The definition of allowable categories of attributes can be performed at the beginning of each execution, or can be defined once and stored on a file and then utilized for any number of executions by giving the name of the file to use at the beginning of an execution.

Figure 1 illustrates how allowable categories of data attributes and their valid values are defined for chains (in this case, land parcels). The allowable categories are as shown in Table I.

Figure 2 illustrates a similar definition but this time for links of street networks.

Both illustrations include the request to file the allowable categories so that they can be used in future executions merely by giving the name of the file with the command COGO.

COGO 'UG2' 'UGCAT'

TABLE I—Allowable Attributes for Parcels

name	meaning	range of values
STREETNO	street number of parcel	any integer value
WARD	ward in which parcel occurs	integer 4, 5 or 9
BUILDING	is there a building on the parcel	YES or NO
TYPE	type of material of the building	BRICK, WOOD, STONE, or CONCRETE
STORIES	number of stories in the building	any integer value
USE	type of use of the building	APARTMENT, MERCANTILE, etc.
MIXED	if mixed use, more detail about it	any alphanumeric value
LAND VALUE	assessed land value	any integer
BUILDING VALUE	assessed building value	any integer
OWNER	name of owner	any alphanumeric
YEAR	year in which assessment was made	any integer

```

DEFINE CATEGORY CHAIN
ADD 1 'STREETNO' INT
ADD 2 'WARD' FIN INT
      4
      5
*      9
ADD 3 'BUILDING' FIN ALPHA 3
      YES
      NO
*
ADD 4 'TYPE' FIN ALPHA 8
      BRICK
      WOOD
      STONE
* CONCRETE
ADD 5 'STORIES' INT
ADD 6 'USE' FIN ALPHA 14
      APARTMENT
      MERCANTILE
      CHURCH
      OFFICE
      SINGLE
      STORE
      FOUNDATION
      SCHOOL
      LODGING HOUSE
      CLUB
      HALL
      DORMITORY
      GARAGE
      VACANT LOT
      PARKING LOT
      POLICE STATION
      FIRE HOUSE
      SUBWAY STATION
      HOTEL
      POST OFFICE
      LIBRARY
      MIXED
*
ADD 7 'MIXED' ALPHA 24
ADD 8 'LAND VALUE' INT
ADD 9 'BUILDING VALUE' INT
ADD 10 'OWNER' ALPHA 32
ADD 11 'YEAR' INTEGER
LIST
FILE
END DEFINE
    
```

Figure 1—Category definition for chains

```

DEFINE CAT LINK
ADD 1 'NAME' ALPHA 12
ADD 2 'TYPE' FIN ALPHA 6
  STREET
  ALLEY
*
ADD 3 'WIDTH' REAL
ADD 4 'LANES' FIN INT
  0
  1
  2
  3
  4
*
ADD 5 'PARKING' FIN ALPHA 5
  LEFT
  RIGHT
  BOTH
*
ADD 6 'LOW NUM' INT
ADD 7 'HIGH NUM' INT
LIST
FILE
END DEFINE

```

Figure 2—Category definition for links

Storage

Actual attribute values for specific objects, e.g., for specific chains, are stored through use of the STORE TEXT command. Figure 3 shows some texts being stored for some land parcels, the allowable categories of which are the same as defined previously (in Figure 1). Once stored, the data attributes can be modified by using the UPDATE TEXT command, deleted via the DELETE TEXT command, summary listed via LIST, or printed by PRINT TEXT.

The attributes can be used for such things as sorting, selective tabulation, creation of statistical sample vectors, selective displaying or plotting, display or plot annotation, and density mapping. Such use is further described in the respective functional sections.

Any number of categories of information may exist for any particular object type and any number of object types may have categories defined for them.

GRAPHICAL INPUT/OUTPUT

Graphical capabilities exist for both input, output, and identification, using a "soft-copy" device (inter-

active display unit) or a "hard-copy" device (flat-bed plotter/digitizer unit) or any combination thereof. The initial work on the system has separated the soft from the hard by essentially assuming that gross sketching and/or gross figure definition is performed on the soft device. These separations are not system imposed but rather decisions made as to the use of the devices dictated by the units themselves, and probably would hold true for most display and plotter units on the market today.

Hardware

The device used in developing the system is shown in Figure 4. It consists of an interactive storage tube graphic display, a keyboard, a printer, a flat-bed plotter/digitizer, and a digitize function keyboard. Probably, the most unique and functionally useful thing about the unit is the fact that the whole set of basic units (i.e., display, keyboard, printer, plotter/digitizer, and function keyboard) are all parts of the same unit, i.e., all controlled by one control unit and require but a single hook-up channel to a computer, be it telephone link to a remote computer or a direct attachment to a local dedicated computer.

```

STORE TEXT CHA '1092'
'STREETNO' 532
'WARD' 4
'BUILDING' 'YES'
'TYPE' 'BRICK'
'STORIES' 4
'USE' 'MERCANTILE'
'LAND VALUE' 385300
'BUILDING VALUE' 914700
'OWNER' 'NEW ENGLAND MUTUAL LIFE INSURANCE COMPANY'
'YEAR' 1964
END
D TEXT CHA '1092-1'
'STREETNO' 508
'WARD' 4
'BUILDING' 'YES'
'TYPE' 'BRICK'
'STORIES' 4
'USE' 'MERCANTILE'
'LAND VALUE' 144000
'BUILDING VALUE' 135000
'OWNER' 'NEW ENGLAND MUTUAL LIFE INSURANCE COMPANY'
'YEAR' 1960
END
D TEXT CHA '1093'
'STREETNO' 490
'WARD' 4
'BUILDING' 'YES'
'TYPE' 'BRICK'
'STORIES' 4
'USE' 'MERCANTILE'
'LAND VALUE' 300000
'BUILDING VALUE' 370000
'OWNER' 'NEW ENGLAND MUTUAL LIFE INSURANCE COMPANY'
'YEAR' 1967
END

```

Figure 3—Storing of textual attributes for chains

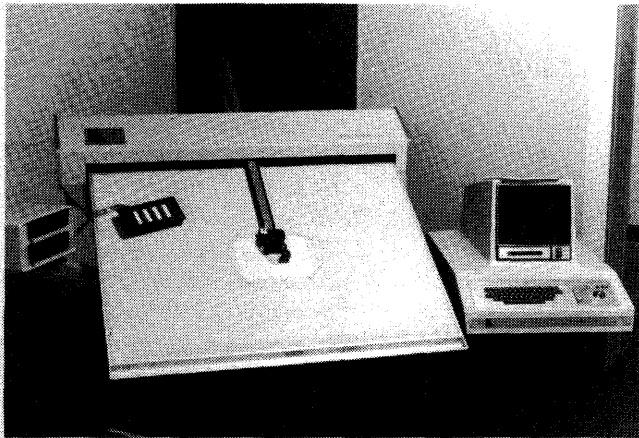


Figure 4—Interactive display and plotter/digitizer

Even though the design and development of the URBAN COGO system and the graphics part in particular has been done using the specific device described above, the system has been designed to permit the use of any such hardware as long as it can be driven in a functionally similar manner. Merely by using a COGO system setting command, and having the appropriate interface programs to physically drive the units, achieves compatibility with other plotters and interactive displays.

```
SET SYSTEM PLOTX 30.0 PLOTY 24.0
UNITP 0.001
```

The set system command can also be used to limit the size of the plot (as above) or of the display, thus permitting different plotter table sizes, different storage tube sizes, and different physical paper sizes on the plotter. The UNITP parameter tells the system what the size of a plotter unit is, in this case, .001 inch.

Graphic input

Inputting of information from the digitizer (flat-bed digitizer) is most powerful.

```
DIGITIZE {
  XY
  XZ
  YZ
} POINT n SCALE
          {
  length
  length PER VALUE
} ORIENTATION
direction DATUM {
  X value Y value
  POINT m
}
```

The digitize command puts the user into digitize mode and sets up basic values to be used in the ensuing storing of the digitized points. n is the point number at which the system is to begin storing points. The SCALE parameters define the scaling to be performed in translating from digitizer coordinates to actual coordinates. It can be given as 200 PER 1 meaning perhaps 200 feet per inch or it can be given as 200 per whatever length is digitized as two points after the command is given. The ORIENTATION defines what rotation is to take place upon the digitized points prior to storing them and can be given as N 0 E where the corresponding direction of due north on the map which is being digitized is defined by 2 points digitized after the command is given.

The last parameter, that of DATUM, defines a base point about which translation, rotation, and scaling are to occur. The values define the actual coordinates (if POINT m option is used, the coordinate values of stored point m are used) of the base point and the digitizer coordinates of that point are defined by digitizing it after the command is given.

Thus, after issuing the command with its appropriate parameters and digitizing 5 or 3 points (5 if SCALE length is given, 3 if length PER value is given—[2 for scale], 2 for orientation, 1 for datum), the system is ready to start accepting digitized information to translate, rotate, and scale it, and then to store it.

Any kind of allowable geometric object from points to chains and points to networks can be defined and stored in this way. To enable the user to tell the system what object or objects he wishes to define, the function keyboard is used. Its template for this command is currently defined as shown in Table II.

TABLE II—Function Key Definitions

Fcn. Key	Action
1	POINT
2	COURSE
3	CURVE
4	CHAIN
5	NETWORK
6	LINK
7	END OF OBJECT
8	END OF OBJECT CLASS
9	PLOT everything digitized up to this point
10	DISPLAY everything digitized up to this point
11	NAME the last object by keying in from keyboard
12	TEXT (or label information) is to be located at next point digitized and the textual information itself entered from the keyboard
16	RETURN to the standard keyboard command environment

TABLE III—Digitizing Sequences

Chain 1	Function key 4	Network Chain 2	Function key 7
	Digitize point		Digitize point
	Digitize point		Function key 7
	Function key 3		Function key 8
	Digitize 5 points		Function key 5
	Function key 7		Function key 6
	Function key 2		Digitize 6 points
	Digitize 2 points		Function key 6
	Function key 7		Digitize 2 points
	Function key 7		Function key 8
	Digitize point		Function key 16
	Function key 3		
	Digitize 3 points		

The functions are hierarchical in that once the chain key is pushed, everything that follows, be it points digitized or combinations of point, curve, course function keys and digitized points, will be stored as one chain until the end of object or end of object class key is pushed. Thus, the sequence in Table III will store 2 chains and 1 network.

The objects digitized in Table III consist of, respectively:

- chain 1: 2 points, 1 curve, 1 course
- chain 2: 1 point, 1 curve, 1 point
- network 1: 5 links composed of 6 points, 1 link composed of 2 points

If no name is given by the user (as above), the system will automatically assign unique names to all objects defined.

The plot and display functions are very useful to the user by enabling him to take spot checks on what he has done so far without removing him from digitize mode.

It has been found that the accuracy obtained from inputting coordinate data in this fashion is entirely dependent upon the amount of time the user wishes to take to accurately position the cross-hair over a position to be digitized. With enough care, accuracy to better than .01 inch can be achieved.

Graphic output

A wide variety of graphical output capabilities exist in the system and will be expanded upon in the future. Straightforward plotting or displaying of geometric objects was the first graphic capability implemented. All objects, points, courses, curves, chains, blocks, regions, of any level, and networks, can be drawn and translated, rotated or magnified. It is thus possible to

display a large section of the city, home in on a particular subarea of interest through one or many repeated magnifications, and then get a hard copy plot of the area of interest. Figures 5 and 6 show results of a display and a plot, respectively, of several blocks of a city.

A display or a plot can be annotated with the name of each object on it, be it all point numbers or all chain names or all block names, etc. The user can also point to an object and have its name or number drawn. Stored attributes about the plotted or displayed objects can also be drawn. Figure 7 shows a block of a city plotted and then annotated with the names of the parcels in the block and the use of each of the parcels as recorded in the assessor's office. The caption "mixed" means that the parcel has mixed usage. Figure 8 shows a plot of what each mixed usage is, drawn so as to use the sheet as an overlay.

A selective draw can also be done, whereby the user requests all of a certain object type whose data attributes satisfy certain criteria to be displayed or plotted. Figures 9 and 10 illustrate this capability. The first figure is a display of all chains in a section of a city whose land value is greater than \$75,000 or whose building value is greater than \$150,000. The second figure is a display of all chains in the same section of a city whose land value is greater than \$75,000 and whose building value is greater than \$150,000.

Just as objects which are drawn in a standard fashion can be annotated, so can objects that are selectively drawn. Figure 11 shows the result of a selective plot and an annotation. The plot requested was all chains in a section of a city whose land value is greater than

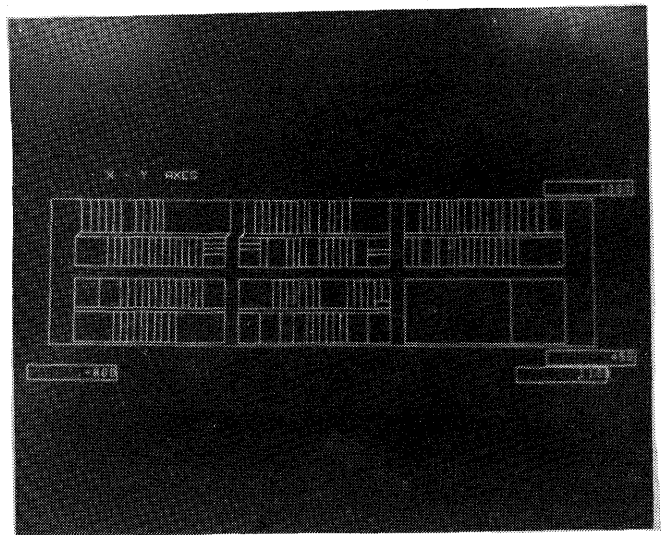






Figure 5—Display of several blocks

\$100,000. That plot was then annotated with the chain names and the land value for each chain.

Another graphic capability is density mapping. Sets of ranges of values for an attribute of chains, blocks, or regions can be drawn, each set with a different shading line. Figure 12 illustrates this by showing different ranges of land value for parcels in a block of a city. The ranges chosen for this were 1,000 to 10,000 designated by , 10,000 to 25,000 designated by , 25,000 to 75,000 by , and 75,000 to 500,000 by . This capability can be used on the display as well as the plotter.

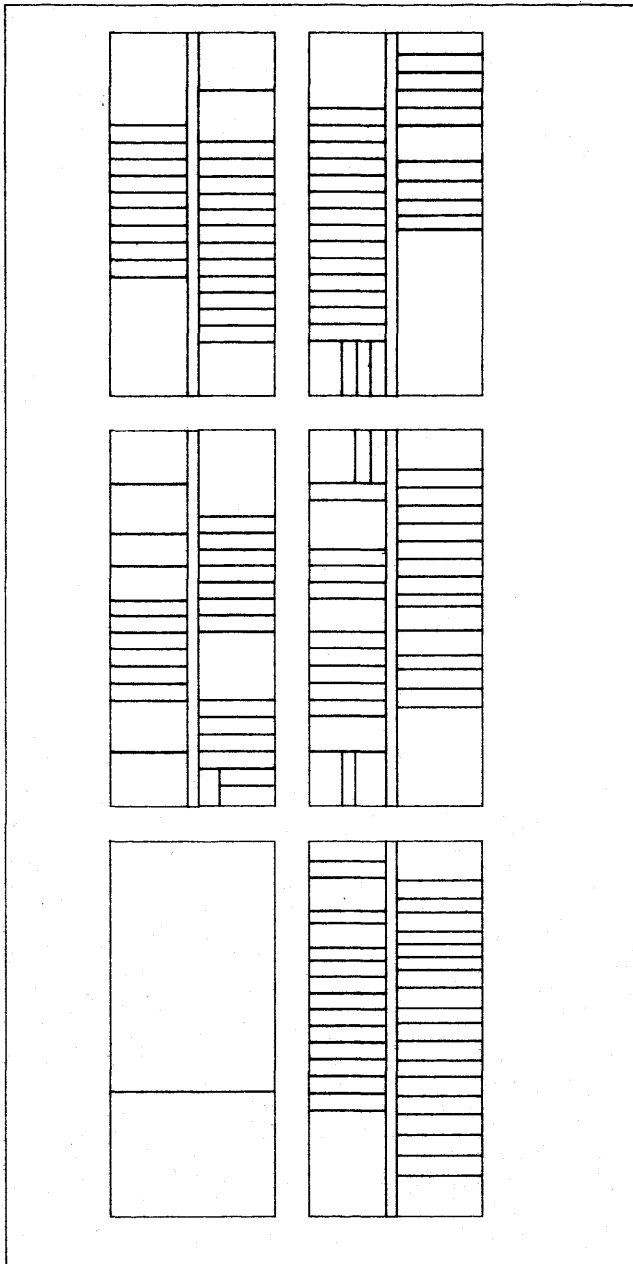


Figure 6—Plot of several blocks

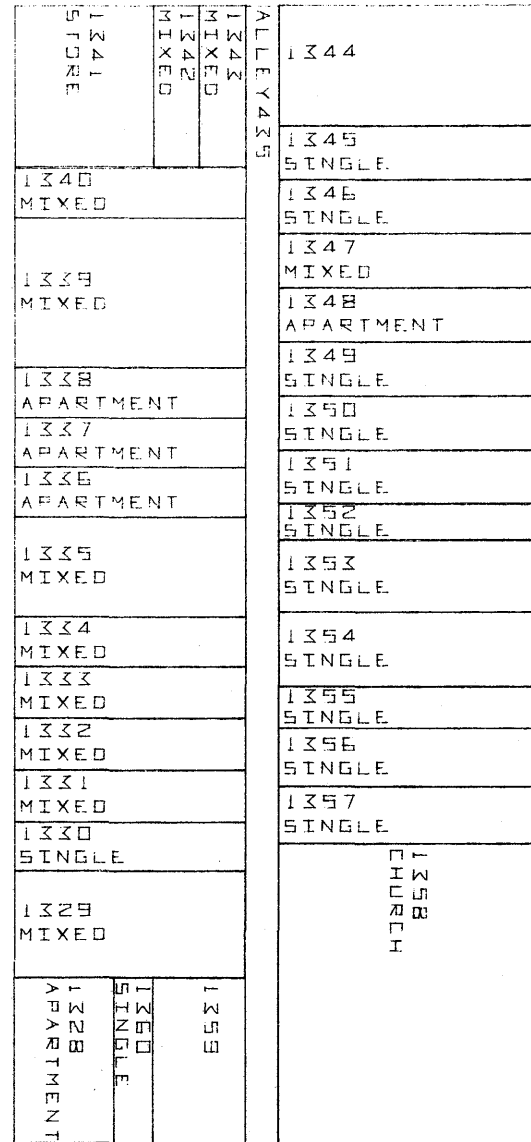


Figure 7—Annotated plot of a city block

Statistical charts and graphs can also be drawn. They are described in the section about the statistical capability of the system.

Additional graphical capabilities are planned for the system but not yet implemented. Full mapping capabilities including full or partial dimensioning and symbol labeling is planned. Examples of such use are tax mapping and utility network schematics. Planar perspectives of three dimensional objects is also planned. Usage for these includes skyline perspectives along a street and multi-tiered transportation and utility networks.

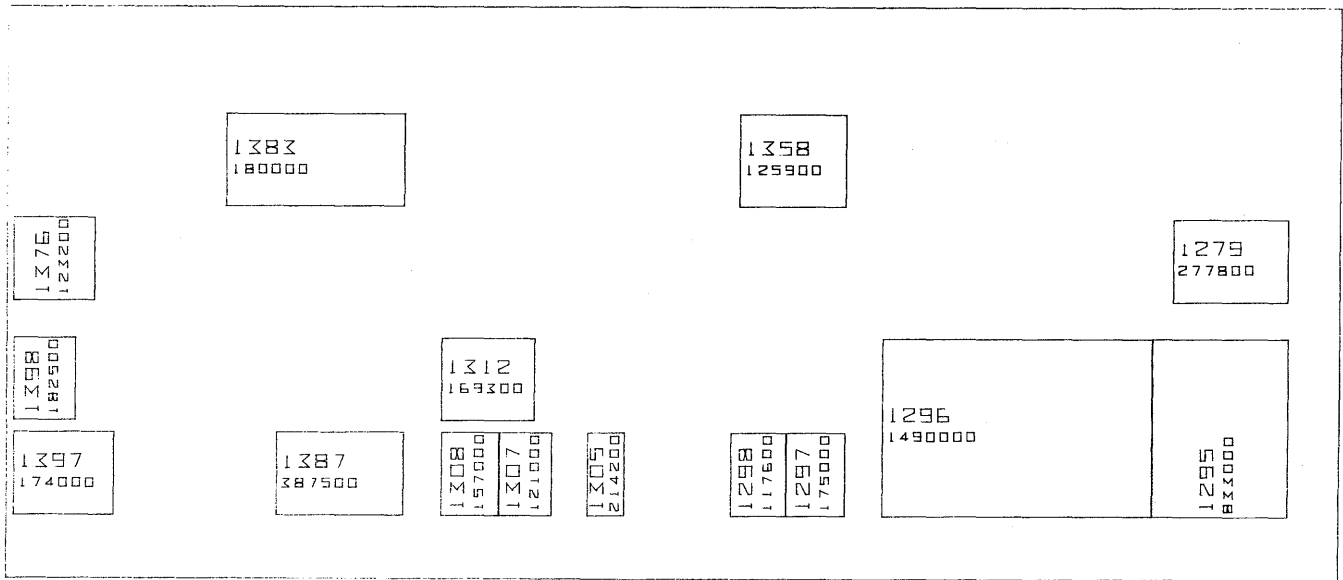
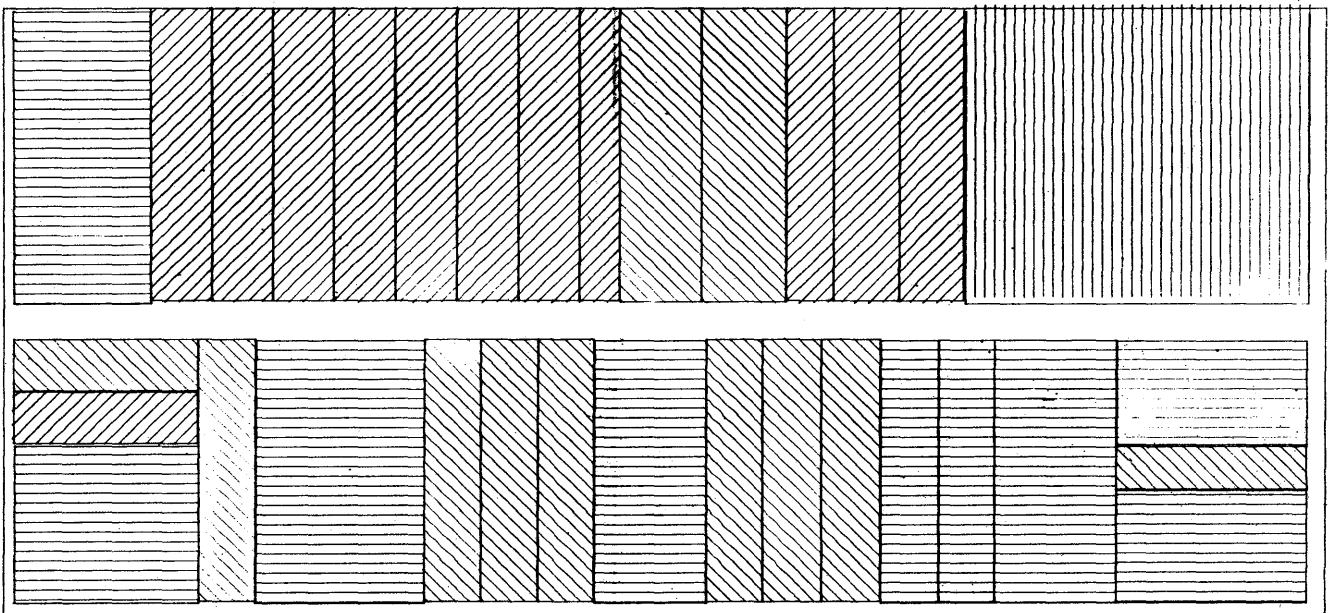


Figure 11—Annotated selective plot

Parametric
 Variance, covariance
 Correlation
 Regression

Using the statistical subset of URBAN COGO, it is possible, through combining, rejecting, creating new vectors with the results of analyses, and so forth, to combine and aggregate in almost any way to any level required, starting at even the most raw level. Capabilities exist for the sample vectors to be saved in a file so that statistical investigation and analysis can be done at the user's leisure.

Additional analyses, such as time series and factor analysis, are planned.



X SCALE 41 PER INCH
 Y SCALE 41 PER INCH

Figure 12—Density plot on land value

The statistical subset also provides the user with the ability to graphically portray a vector or group of vectors. Graphs, scatter diagrams, and histograms can be drawn on the printer, the display, or the plotter. Figures 13 and 14 illustrate the plotting of a graph and a histogram, respectively. The first is a graph of the land values of all parcels in a section of a city. The second is a histogram of the use of the parcels in a section of a city.

PROCESSING FUNCTIONS

Various processing facilities have been included. ICES COGO contained the ability to locate points in the XY plane in many ways, including:

- at a given distance and direction from another point
- at the intersection of 2 lines
- at the intersection of 2 curves
- at the intersection of a curve and a line
- as a projection onto a course or a curve
- at a given distance along a course or curve

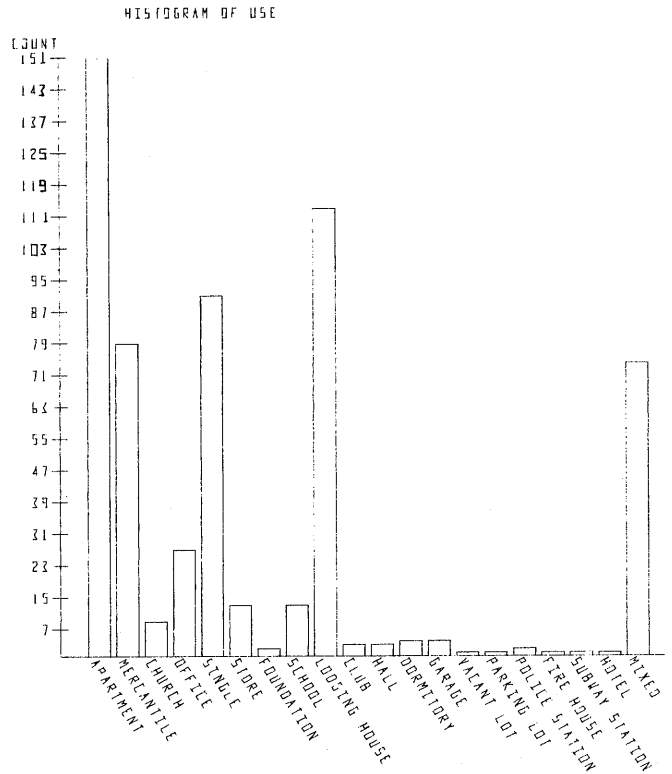


Figure 14—Histogram plot on usage

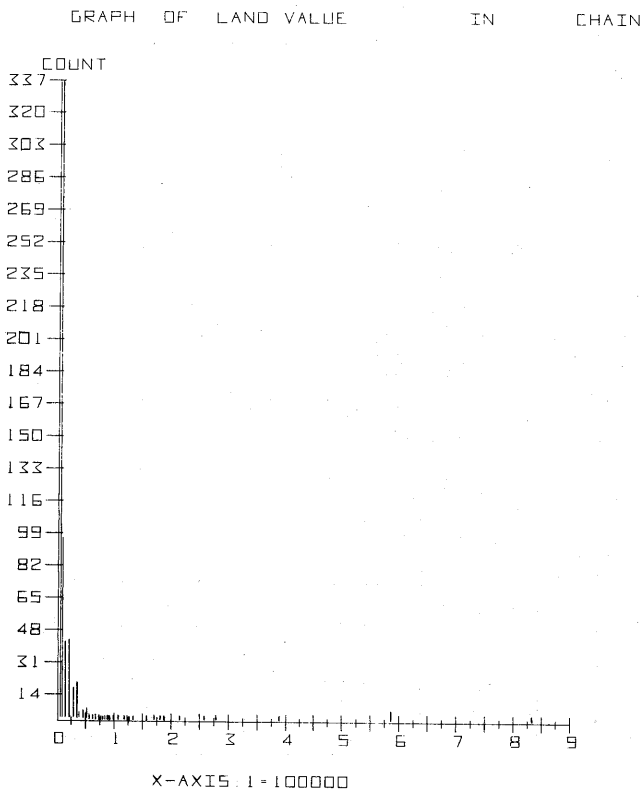


Figure 13—Statistical graph plot on land value

URBAN COGO has expanded upon this to enable the user to intersect higher level objects and store the points of intersection as new points. Networks, chains, and blocks can all be intersected with similar objects, e.g., network with network, or with other objects, e.g., network with block.

It is planned to expand upon the location capability as well as the intersect capability to move along any planar slice in 3-space when the three-dimensional object capability is implemented.

Translation and rotation of networks, chains, and blocks are also provided with the ability to do so on any of the three standard planes—XY, XZ, or YZ. Thus, with this capability, a user may store groups of objects in their own local coordinate system and when they are fully checked, translate and rotate them to a more global coordinate system.

Tabulation

Tabulation facilities are also available and can easily be added to for any specific use of the system. General capabilities such as selective tabulation of objects whose data attributes satisfy certain criteria, or the sorting of

a class of objects on one or more data attributes are now part of the system. Major report generation could easily be added to the system, but is highly user and type-of-data dependent. An illustration of one such report is the generation of tax bills for the assessors office. Adding this capability would be very easy once the format for the bills has been defined. It is easy to foresee many facilities being added to the system in the area of report generation, but primarily by and for a specific set of users.

CENSUS DATA

Many potential users of URBAN COGO will want to interface with census programs and to use census data. Commands and their associated programs are being developed to be able to retrieve any or all of the data on a census tape and store it in a defined way in the URBAN COGO data base. This facility is being developed so that not only census data can be retrieved and used in this way but almost any kind of data that a user might want to utilize in the system.

The retrieving and the defining of the correspondences between the original data and its URBAN COGO counterpart is structured so that attribute data as well as geo-data (such as that used in the DIME system) can be handled.

It is foreseen that this capability will be one major way in which users will be able to interface other systems with URBAN COGO.

EDITING

There is one very important fall-out of the work done to date on the system, namely the ability of the system capabilities to be used for data editing, data error detection, and self-checking. The features which have proved very useful for error detection are:

- the definition of allowable categories of data and the allowable values these attributes may have;
- the display and plot facilities to catch errors in the definition of geographic objects;
- the sorting and statistical tabulation commands to find objects for which no data is stored or for which only partial data is stored.

The fact that a user can essentially use the system to check the system is of major import, and the fact that it is provided as fall-out (by design) is also of major importance. It is most likely, however, that this fact will only be recognized through use.

USAGE SAMPLING

The potential uses for and users of the system are many and varied. A brief summary of some follows:

- City Government: tax billing, analyses of effects of tax changes, mapping, management reporting, area redevelopment;
- State Government: transportation route location and analysis, congressional districting, maintenance of highways and highway signs;
- Utility Companies: maintenance, location of manholes, prime power supplies, utility network mapping;
- Consulting Companies: transportation studies, airport location studies, urban planning;
- Legal Services: title searching.

The specific uses of the system are potentially too numerous and varied to describe. Perhaps the best source is the imagination of each user.

SUMMARY

While it is believed that the URBAN COGO system described above provides the base for a new kind of urban information system, it is also believed that much work still remains to be done with it in order to prove this belief. Two major areas still remain to be researched.

- the design and development of a new file structure
- the use of the system in some specific applications

This work is planned to be performed in the coming year.

The author believes that the building of the system upon geographic data as its base and the heavy emphasis of graphical capabilities in the system will prove to be the way-of-the-future for urban information systems.

ACKNOWLEDGMENTS

The URBAN COGO system reported on above is being developed by the Urban Geometrics project of the Urban Systems Laboratory of the Massachusetts Institute of Technology. The author would like to acknowledge the advice and wisdom of Professor C. L. Miller, Director of the Laboratory, who originated the COGO system and who formulated the original concept of URBAN COGO. The author would also like to acknowledge those M.I.T. students (all of whom are members of the Chi Phi Fraternity) who are and have worked hard and diligently on making the system a reality.

The URBAN COGO project is currently being supported, in part, by a grant from the National Science Foundation. Previously, the project was sponsored in part by grants to the Urban Systems Laboratory from the Ford Foundation and the IBM Corporation.

BIBLIOGRAPHY

- 1 A J CASNER W B LEACH
Cases and text on property
Little Brown and Company Boston 1969
- 2 F E CLARK
A treatise on the law of surveying and boundaries
The Bobbs-Merrill Company Indianapolis 1939
- 3 M CLAWSON C L STEWART
Land use information
Resources for the Future Inc The Johns Hopkins Press
Baltimore 1965
- 4 R T HOWE
Fundamentals of a modern system of land parcel records
Department of Civil Engineering University of
Cincinnati May 1968
- 5 LOCKHEED MISSILES AND SPACE COMPANY
California statewide information system study
Sunnyvale California July 1965
- 6 C L MILLER
Engineers' guide to ICES-COGO I
Department of Civil Engineering Report No R67-46
Massachusetts Institute of Technology August 1967
- 7 B SCHUMACKER
An introduction to ICES
Department of Civil Engineering Report No R67-47
Massachusetts Institute of Technology 1967
- 8 B SCHUMACKER
URBAN COGO users' guide
Urban Systems Laboratory, Massachusetts Institute of
Technology June 1970
- 9 *Census use study reports*
US Bureau of the Census Reports Nos 1 through 11
Washington DC 1970
- 10 _____
*Urban and regional information systems: support for
planning in metropolitan areas*
US Department of Housing and Urban Development
Washington DC October 1968
- 11 _____
Operational and maintenance manual for interactgraphic 1
Computervision Corporation Burlington Massachusetts
October 1970

Understanding *Urban Dynamics**

by GERALD O. BARNEY

Center for Naval Analyses
Arlington, Virginia

INTRODUCTION

As indicated by published reviews and unpublished criticisms, some readers have had difficulty in understanding several of the most important points of *Urban Dynamics*** by Professor Jay W. Forrester. The book contains several stumbling blocks. For example, certain pet theories that for years have been thought to be important in the dynamics of an urban area are scarcely even mentioned (e.g., transportation, crime, pollution, discrimination and suburbs). Also, several measures of urban characteristics appear to be sufficiently different in the model from those found in real urban areas to distract one's attention from the main points of the book. But there is a message in *Urban Dynamics*, and when it is comprehended, these stumbling blocks become less significant. This paper is intended to help the reader of *Urban Dynamics* to understand the message of the book and to see beyond many of the criticisms that have been made.

WHAT IS *URBAN DYNAMICS*?

Urban Dynamics is an analysis of how the urban system operates and how it can be more effectively managed. The development of large concentrations of relatively unskilled persons and the blighting effect these concentrations have on our people and cities are the primary issues discussed in the book. The analysis is based on a computer model which, in the most general terms, simulates the interactions among population, housing, employment (industry) and municipal services.

The urban system is an example of what has become

known as a "complex system"—a system whose behavior is dominated by multiple-loop, non-linear feedback processes. Mathematical analysis is not too helpful in understanding complex systems since their non-linear properties are as yet very difficult to treat analytically. Currently, the only successful method of dealing with systems as complex as the urban system is experimentation—with the actual system or with some representation of the actual system. In the case of the urban system, most of the experimentation is done with a mental representation—the mental image (or model) we each have of how the urban system operates.

Our public officials are constantly performing experiments with their mental models as they evaluate proposed changes and additions to laws and policies. Although most public officials are probably not explicitly aware of it, their experiments involve three separate and distinct steps. The official first brings to mind his latest mental image of how the system operates; he then uses his mental model to deduce the effects of the proposal; and finally he judges his deduction of the effects against his set of values and goals. In the past, it has not been too important to distinguish these three steps, but as the policy and legislative issues become more complex, it is increasingly important to know whether disagreements over a given proposal stem from different conceptions of how the system works, from inaccurate or inconsistent deductions of effects, or from more basic differences of values and goals.

In turning to the computer for assistance, we are forced to consider each step separately. Our mental image must be developed and expressed in a language that can be used to instruct the computer. Any consistent, explicit mental image of any system can be so expressed. Our mental images are the results of our experiences, and expressing these experiences explicitly for the computer permits others to examine, correct, and comprehend our mental images and to contribute to a broader understanding through their different experiences. Given the expression of our mental image,

* Dr. Barney is employed by the Center for Naval Analyses of the University of Rochester, 1401 Wilson Boulevard, Arlington, Virginia 22209. This paper was written while Dr. Barney was on leave at the Massachusetts Institute of Technology.

** Published by the MIT Press, Cambridge, Massachusetts, 1969.

the computer can point out inconsistencies, determine sensitivities, and deduce implications much more accurately than can the human mind—and without changing the ground rules part way along as the human mind is so prone to do.

But probably the most important contribution the computer makes is that it forces us to give separate consideration to questions of values and goals. Given the implications of a proposed change in a law or policy, we are forced to ask if this is what we want, if this is consistent with our values, and if this brings us any nearer our collective goals. With finite resources, cities can't be everything to everyone. Given a better understanding of the options available and the effects of any given proposal, debate must then center on the desirability of the effects and the values necessary for judging desirability.

By passing laws and changing policies, our public officials are making changes in the very structure of our society. To be of assistance in the analysis of the questions they face, a model must not only reproduce the behavior of a city in a general sense, it must also correctly reflect the basic causal mechanisms at work. Many basically different models can reproduce urban history, but a model that is to be used to examine the effects of change in structure must correctly reflect all of the important causal mechanisms—some of which are not yet easily measured. This is a formidable requirement, and success for now must be measured not against an absolute standard of accuracy but rather against our only alternatives—inexplicit mental models or intuition.

THE HEART OF THE MODEL

The Urban Dynamics Model is an explicit expression of a distillation of several mental images. Its subject is the causes of urban decay—the concentration of large numbers of relatively unskilled people in urban areas, and all the attendant problems. In the model, just as in real cities, people migrate in and out and move among the socio-economic classes (Forrester defines three such classes: Underemployed, Labor, and Management-Professional) in response to a variety of conditions, including population, housing, employment, and municipal services. The heart of the model is the enumeration and description of the multitude of urban conditions which influence migration and economic advancement.

The concept of “attractiveness,” the central idea behind Forrester’s description of migration, is frequently misunderstood. Attractiveness is not an indication of a city’s beauty but rather a measure of a city’s drawing

and holding power for people in the three socio-economic classes. Although a city’s attractiveness is generally different for the Underemployed, Labor, and Management-Professional populations, there is an attractiveness for each of the three classes determining the rates at which they are drawn to the city and how effective the city is in holding them there once they arrive. Forrester inadvertently confuses many readers when he gives the attractiveness indicators the following three apparently unrelated names: Attractiveness for Migration Multiplier (AMM), Labor Arrival Multiplier (LAM), and Management Arrival Multiplier (MAM). The identical modeling function of the three attractiveness multipliers is indicated in Figure 1.

The attractiveness multipliers are especially important in that they reflect the population’s response to a variety of “incommensurables.” Population, housing, housing programs, economic advancement potential, public expenditures and employment opportunities are reduced to a common scale or commensurated (differently for the three classes) to give a composite attractiveness for each class. For example, attractiveness for the Underemployed population increases with increased

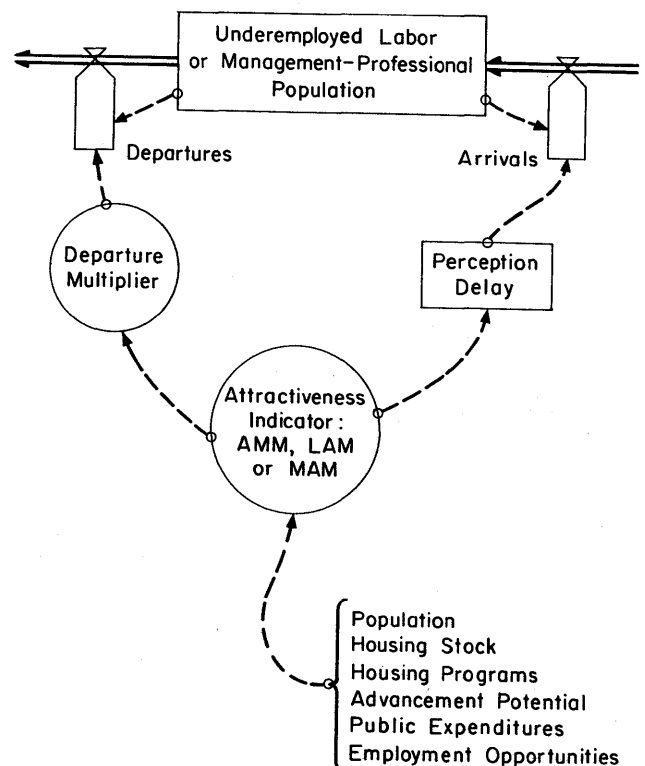


Figure 1—A flow diagram summarizing how the attractiveness indicators are used to influence the arrivals and departures for the three socio-economic classes (see *Urban Dynamics*, pages 134, 160 and 165)

housing programs, economic advancement potential, public expenditures, and employment opportunities, but decreases with increased Underemployed population (reflecting more competition for jobs, housing, etc.). Wherever differences in attractiveness exist, the population gradually migrates to the more attractive area, and the changed population distribution gradually reduces the difference in attractiveness.

Significant differences in attractiveness can exist only across boundaries where migration is restricted (e.g., between cities in Mexico and California). Within the United States, however, attractiveness is essentially constant. When all components of attractiveness are considered, New York City, Chicago, Colorado Springs, and Bend (Oregon) have very nearly equal attractiveness for a given socio-economic class.

Another important part of the model is the description of the factors that determine how fast people advance from Underemployed to Labor and from Labor to Management-Professional. The rate of advancement from Underemployed to Labor (UTL) is particularly important, since it is only through this transition that the Underemployed can escape poverty. The conditions that influence UTL are total labor and underemployed jobs, Labor and Underemployed populations, education level of Underemployed, job training programs, and the ratio of Labor (teachers) to Underemployed (students).

THE FAILURE OF CURRENT URBAN PROGRAMS

The importance of the advancement and attractiveness concepts can be seen in Forrester's analysis of current urban programs. In actual practice, these programs generally have a similar and characteristic development pattern: an initial period of slight improvement and generation of hope, followed within a few years by readjustments within the urban system which result in a loss of gained ground and general disillusionment of the Underemployed. The net result has been increased concentrations of Underemployed, continually decaying conditions, and growing hostility of the Underemployed toward the "System" and toward the "Establishment" that they think controls the "System." Actually, as Forrester's analysis shows, the failure of our urban programs is due not to the control of the establishment, but rather to a collection of feedback processes that are at work within the system and are almost beyond the influence of the establishment. The advancement and attractiveness concepts are important in understanding the operation and effects of these feedback processes.

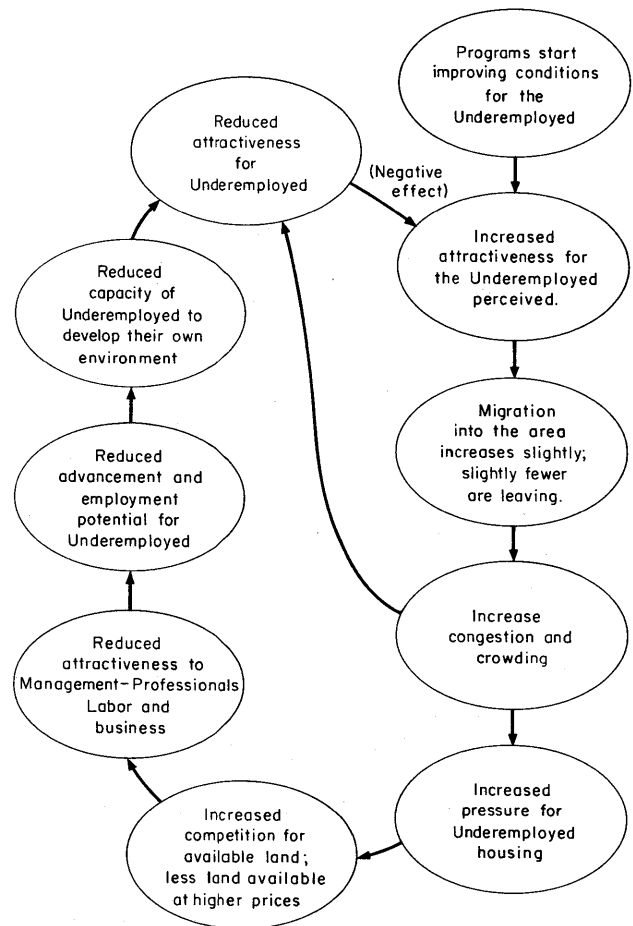


Figure 2—Illustration of two negative feedback loops which tend to undermine the effects of direct aid to the underemployed

There are many interacting feedback processes that cause urban problems to feed on themselves and make failures of our urban programs, but in a highly simplified way, two of the most dominant interactions are illustrated in Figure 2. Shortly after the initiation of any given program (housing, food, health, job training, etc.), conditions for the Underemployed do measurably improve, and the improvement encourages continuation of the program. In time, the increased attractiveness of the area is evident to the Underemployed, and, as a result, a somewhat larger number move into the area and a somewhat smaller number leave than would have, had the attractiveness not increased. There follows a period of somewhat expanded growth of the Underemployed population in the area, and this population growth increases the pressures on the available schools, housing, employment opportunities, shopping and recreational facilities, and transportation systems. The effects of this first feedback loop are felt

within a few years when the increased crowding and congestion begin to drop the total attractiveness back toward what it was before the program was started.

The effects of the second feedback loop are delayed by another few years. The enlarged Underemployment population, which resulted from the initial success of the program, increases the demand for Underemployed housing. As a result, Underemployed housing competes more and more vigorously for available land. This competition not only takes vacant land but preserves very decayed housing stock that might otherwise be destroyed to permit an alternative land use. The increased demand for Underemployed-housing land-use drives up the cost of land for Labor and Management-Professional housing—thus reducing the attractiveness of the area for these two populations. This reduced attractiveness for Labor and Management-Professionals, combined with the more intense competition for land, implies more business expenses, fewer business opportunities, and increased difficulties for all forms of business activity. Declining business activity reduces the advancement potential for the Underemployed. The lowered advancement potential in turn diminishes the chances of the Underemployed escaping from the urban poverty trap, destroys their chances of improving their living conditions, and increases their frustration and hostility.

THE PROBLEM AND FORRESTER'S SOLUTION

The really basic problem is this: Given the existence of feedback processes which tend to counter the effects of direct improvements, how can a city best improve the lot of its Underemployed with the limited resources it has available? Forrester's approach to this problem is significantly different from many approaches used in the past. He does not start by stating what cities should be like but rather asks the very practical question: In what ways can the urban system be made to operate differently? The effects of changing the way the system works are then investigated with the model, and the "best" alternative is recommended.

But "best" for whom: the rich, the poor, the absentee landlords, the stock brokers? What values are to be used in judging the possibilities? Forrester doesn't explicitly discuss the values he uses, but implicit in his discussions of the alternatives are two values: the solution must be lasting (decades at least) as opposed to temporary (a few months to a few years), and the solution must lead to increased upward economic mobility for the Underemployed. Others (slumlords, for example) might have different objectives and values to use in evaluating the results of the computer runs,

but the values behind Forrester's discussions deserve careful consideration. Temporary solutions have produced much disillusionment, frustration, and resentment among the urban poor, and while the poor may not be able to agree on which particular urban conditions are most in need of improvement, increased upward economic mobility provides them with both hope and freedom of choice.

Forrester's choice for the best way to revive a decayed urban area is to demolish five percent of the slum housing stock (some of which would have been destroyed anyway and much of which is already vacant) each year and to provide business encouragements that increase new enterprise construction 40 percent over what would have occurred under the same conditions without the added incentives. The demolition need not involve active intervention by the city; it is probably best accomplished through changes in tax laws and zoning. The increased availability of land improves the attractiveness of the area to business, Labor, and Management-Professionals. This in turn results in an upsurge in the demand for labor, which in turn increases the opportunities for upward economic advancement for the Underemployed. As the area becomes more of a place for the Underemployed to get ahead, its attractiveness to the Underemployed begins to increase, and if unchecked, a new Underemployed immigration (and a resulting demand for Underemployed housing) would within a few years increase competition for the available land to the point that business opportunities (and the associated advancement potential for the Underemployed) would again be decreased. In Forrester's solution,^{***} attractiveness and migration are limited by reduced housing stock available to the Underemployed.

COMMON MISUNDERSTANDINGS

Forrester's proposed program of slum housing demolition provides an interesting demonstration of need for more than intuition in predicting the dynamics of urban systems. Intuitively it would seem that slum housing demolition could do nothing but make housing scarcer and ultimately drive the Underemployed out of the area. At first glance, the analysis from the model also seems to support these conclusions, in that the ratio of Underemployed to Underemployed-housing increases significantly. But something else is happening. The net immigration of Underemployed increases by almost 4,500 persons per year, a factor of approximately 450. Why? Because even though housing is

^{***} See *Urban Dynamics*, Section 5.7.

tighter, the increased business activity and potential for economic advancement actually make the area very attractive to the Underemployed.

It seems paradoxical that reduced housing makes it possible for more Underemployed to migrate into the area. In spite of the increased influx, the Underemployed population actually decreases by 10 percent because many more are advancing into the Labor population. The net annual-advancement-rate from Underemployed to Labor rises from about 5,500 to just over 9,000 persons per year, 165 percent of the old rate. In contrast to the urban renewal programs of the fifties, this program works slowly and does not completely disrupt and destroy whole communities. In Figure 3, the effects of the program on population movements to, from, and within the urban area are illustrated. In addition to being economically viable, the area is now an efficient upgrader of the population. The overall effects are indicated in Table I.

TABLE I—Changes in the Population Mix and Land Use Distribution Resulting from Slum-housing Demolition and Industry Encouragement

	Equilibrium ¹ Mode	Revival ² Mode	Percent Difference
Land used for income-producing activities (acres)	5,800	8,600	+50%
Land used for housing (acres)	75,700	77,900	+ 3%
Ratio of housing-land to income-producing land	12.96	9.05	-30%
Management-Professional population	71,000	109,000	+50%
Labor population	393,000	600,000	+50%
Underemployed population	377,000	336,000	-10%
Total population	341,000	1,045,000	+200%

¹ This is the starting-point for all runs in chapter five.
² See chapter five, section 5.7.

OMISSIONS?

The *Urban Dynamics* work has been criticized for the omission of a variety of factors that are alleged to be central to the urban problem. Influences of the suburbs,

transportation systems, discrimination, pollution, and “external driving forces” are among the factors frequently cited. Although some of these factors are very important to certain urban phenomena, the urban dynamics model already contains enough detail to produce the urban phenomenon about which the book is written: the concentration of large numbers of relatively unskilled people in decaying sections of our cities. Additional details are likely to make an already complex model more confusing, and unless they can be shown to be required to produce the problem under study, extra details are probably best left out. Some comments on several of the frequently-noted “omissions” are given in the following paragraphs.

First, *Urban Dynamics* does not assume (as has been asserted) that the dynamics of urban areas are independent of depressions, world wars, technological change, earthquakes, and other “external driving forces.” What *Urban Dynamics* does assume is that the management of a given urban area has little or no influence on the external forces acting on the area, and that, come what may, urban areas must be managed as effectively as possible. Although uncertainties may dictate a more or less cautious advance, *Urban Dynamics* asserts that effective urban management is possible in spite of uncertainties. The effectiveness of urban management seems to be limited not so much by an inability to predict the future course of external driving forces as by an inadequate understanding of the time-dependent consequences of the many non-linear feedback processes at work.

Concerning suburbs and their effects, it should be noted that every city is in competition with its environ-

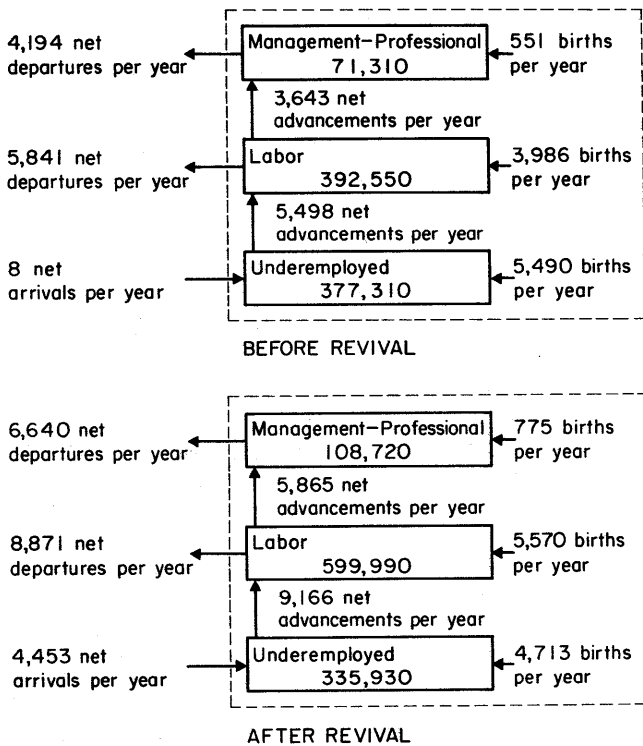


Figure 3—Equilibrium population flows before and after revival. The dashed lines represent the boundary of the urban area

ment (that is, the remainder of the nation) for people and industry. A city's suburbs are a part of its environment, and this basic competitive influence of the suburbs on a city is included in the attractiveness and migration concepts of the model. A city's suburbs are different from the remainder of its environment only in that they are close enough to the city to allow suburbanites to commute to jobs in the city without actually living in the city.

The definitions of the attractiveness indicators and the system boundary are closely related to the suburbs question. In defining the term "urban area" and in specifying the system boundary, *Urban Dynamics* assumes that the population both lives and works in the "urban area" inside the system boundary. In determining the migration rates, the attractiveness of the area as a place to work is not differentiated from the attractiveness of the area as a place to live. Daytime and nighttime populations are equal. In this approximation, some very important effects can and have been studied. It is interesting to note, however, that the conditions resulting from Forrester's solution (good business activity, many job opportunities, a shortage of housing—especially for labor (Labor-to-Housing Ratio = 1.332)) are exactly the conditions that lead to tremendous highway-expansion programs and suburban growth. An expanded suburban population that is allowed to enjoy the attractiveness of the city as a place to work and to enjoy the attractiveness of the suburbs as a place to live would probably have a significant impact on Forrester's recommended solution. Without expansion, the model cannot analyze this impact. The book does, however, suggest ways of minimizing the effect.

Although racial discrimination has declined during the last decade, it is still a significant problem for many Americans—but not their only (and perhaps not even their most serious) problem. But through preoccupation with the discrimination issue, there is danger of losing sight of a basic obstacle to the economic recovery of the victims of discrimination. Discrimination in the past has put many more blacks than whites into the Underemployed category, and now this imbalance makes the basic problem of the Underemployed appear to be a problem of discrimination. But the problems of Watts and Harlem have common roots with the problems of Appalachia. The Underemployed—black or white—are trapped by the same basic mechanism. If today we could completely eliminate all discrimination (and who could deny that a significant amount still exists), the Underemployed black's problem would not be solved. As an Underemployed person, the "System" would keep him right where he is in the inner city ghetto,

with virtually no hope of escape. *Urban Dynamics* describes the basic obstacle to his advancement and indicates how, and at what cost, the "System" can be made to work for the Underemployed, black and white. Until the feedback processes at work in the System are better understood by our leaders and by the general public, many obstacles to the advancement of Underemployed blacks will continue to be easily confused with discrimination.

Although it cannot be called an omission, the "infinite environment" approximation has concerned many people. The urban area of the model is located in an "environment" which acts as an infinite source (or sink) for people of the three socio-economic classes. Migration rates depend only on the relative attractiveness. The attractiveness of the environment is independent of the number of people that are drawn from it or added to it. More specifically, this approximation enters the model through the assumption that, on an annual basis, an urban area for periods of at least 50 years can:

- (1) draw the equivalent of 1 percent of its Underemployed population from its "environment,"
- (2) deposit into its "environment" about 1.5 percent of its Labor population,
- (3) deposit into its "environment" about 6 percent of its Management-Professional population,

without significantly altering conditions in the "environment." One city, critics argue, can probably do this, but if all cities were doing this (as they might if Forrester's solutions were adopted as federal policy) conditions in the "environment" would change. It is then argued that the ensuing changes in relative attractiveness would lead to different conditions than those suggested in Forrester's analysis.

Since the departing Labor and Management-Professional population could well be used in starting new communities (which will be needed if our population continues to grow as assumed in the model), the most likely change in the environment would be a drying up of the source of Underemployed. This seems unlikely to destroy the usefulness of Forrester's suggestion.

THE MESSAGE

In spite of its first appearance, much of what *Urban Dynamics* says closely resembles what urban experts have been saying for years. There is nearly complete agreement, for example, that the fundamental characteristics of a decayed urban area are an inappropriate population mix and an economically unsatisfactory

distribution of land use. Without Managers and Professionals to recognize opportunities and to organize income producing activities, and without a large Labor population from which skills can be learned, it is not surprising that the economic advancement of the Underemployed living in decayed urban areas is rather limited. Yet relatively inexpensive shelter, welfare income, televised entertainment, public transportation, and municipal services attract the Underemployed into our urban poverty traps. As the number of Underemployed in an area increases, many small changes interact to shift land use toward housing and away from income-producing activities. This is another way of describing what Forrester calls excess housing. This mode of operation in which problems feed upon themselves and in which the Underemployed are trapped for generations is widely agreed to be The Urban Problem.

The new and important contributions that *Urban Dynamics* makes are in three areas. First, it describes the basic characteristics of the urban system which cause decay to feed on itself. This phenomena is complex and not easily summarized, but a particularly important aspect of it is the close coupling between land use and population mix. As is very clearly illustrated in the book, neither land-use nor population-mix can be managed independently. A change in one always produces a change in the other. Municipal responsibility for land-use management has long been recognized, but the fact that every land-management and municipal-service decision affects the relative attractiveness of the area to the various socio-economic elements of the population (and thus determines the population mix) has only rarely been openly discussed. *Urban Dynamics* discusses this point quite openly and points out how land-use policies, tax laws, assessment practices and zoning procedures play a major role in

bringing together the population-mix we find in our slums.

A second and related contribution is the analysis of the failure of past urban programs to achieve any lasting impact on the basic problem. Urban renewal (as practiced in the fifties) and the relocation of Underemployed in low-rent suburban housing have either completely destroyed and replaced a community or transplanted the Underemployed to a new location where they are needed and wanted no more than they were before the relocation. These brute force solutions do not recognize and deal with the basic causes of the difficulties and can lead to nothing more productive than localized temporary improvements.

The final and most significant contribution of *Urban Dynamics* is that it provides an approach through which even a single city, acting alone, can make a lasting and significant impact on the distribution of land-use and the population-mix in its blighted urban areas. The solution does not involve pushing the Underemployed out but rather gradually attracting Management, Labor, and business back. This approach requires patience, tenacity and understanding, but it treats the problem rather than the symptoms. Lasting solutions will be achieved only when the underlying feedback processes are recognized and dealt with. Some basic changes and new responsibilities for municipal management are required, but only if we establish as our goal the rebalancing of both the population-mix and the distribution of land-use to maximize the upward economic mobility of the Underemployed, can we hope to eliminate the frustrations of our inner-city Underemployed and the explosive atmosphere that accompanies their disillusionment. In that cities have open to them an effective, independent, and imperative course, they are truly "masters of their own fate."

Bankmod—An interactive decision aid for banks

by WOLFGANG P. HOEHENWARTER and KENNETH E. REICH

Bank Administration Institute
Park Ridge, Illinois

INTRODUCTION

Use of mathematical modeling techniques to assist bank management in managing the sources and uses of funds has become the subject of increasing bank research effort. This emphasis on use of modeling techniques stems from a concern that planning and managing for a sustained increase in bank profits is becoming increasingly critical.

A recent study¹ showed that while bank profits tripled in the decade 1958-1968, much of this increase is due to unusual factors such as rising interest rates, reduction in excess liquidity and the impact of now discontinued accounting practices. The improvements resulting from these factors will not be available to the industry for future profit growth. In addition, this study and others^{2,3} forecast that the nature of credit demand and deposit supply will also change significantly. To maintain profitability, bank management will therefore have to revise their concepts about balance sheet structure and utilize techniques which permit banks to assume more risk in their mix of assets and liabilities. Furthermore, they need to improve the profit margin on funds managed through more opportunistic lending and investment policies and more sophisticated funds management techniques.¹

For these reasons, Bank Administration Institute has undertaken a multi-stage project to develop a series of balance sheet management models. The initial prototype, now in the testing and demonstration phase, is described in this paper. Following an introductory discussion of decision-making problems in banks, the paper outlines the approach BAI has taken and describes the major features of the model, including the handling of the man-machine interface.

CLASSIFICATION OF MANAGEMENT PROBLEMS FACING BANKS

Managing a bank consists largely of continually determining the mix of sources and uses of the bank's

funds in order to maximize long term performance while simultaneously satisfying certain constraints⁴ of the regulatory agencies. These constraints are designed primarily to protect depositors. From the stockholder's perspective, future performance is defined in terms of the expected future return and the uncertainty associated with that return. Stated another way, management must choose the appropriate combination of return and risk which maximizes performance from the view of the stockholders. To achieve this optimum trade-off between risk and return, management must make a variety of decisions which directly or indirectly affect the mix of sources and uses of funds.

For purposes of applying modeling techniques, decisions can be classified into three types:

Long term strategic decisions

These relate primarily to the overall level and direction of the bank's business. Examples include:

- Introduction of major new services
- Expansion of physical facilities
- Implementation of marketing programs
- Recruitment and development of personnel

Major decisions in these areas tend to be made relatively infrequently, have a gradual impact on bank performance and are usually made only after lengthy, in-depth studies. It would be desirable to have models to aid in these decisions as they substantially determine the ultimate growth and profitability of the bank. However, the large amount of necessary information required, the difficulties in quantifying the interrelationships among the many variables, and the uncertainties involved make this task extremely difficult.

Intermediate term balance sheet management decisions

These decisions relate to average levels of sources and uses of discretionary funds for periods of a month

or more in length and a planning horizon of perhaps one to two years into the future. Examples include:

- Balancing anticipated sources and uses of funds to meet liquidity and capital adequacy constraints while maximizing profitability.
- Allocating funds between loan and security portfolios and, within these portfolios, among asset units of various types, maturities and yields.
- Determining appropriate adjustments to make if actual flows of sources and uses of funds deviate significantly from the expected.

These decisions are made relatively frequently and have both an immediate and long term impact on bank performance. In the long run, the overall yield on assets held and the cost of funds used are affected. In the short run, a decision to reallocate assets could result in selling securities at a substantial gain or loss thereby drastically affecting earnings for that reporting period. In contrast to the problems of modeling long term strategic decisions, these decisions require relatively less information, which in turn is much less resistant to quantification.

Short term money management

If balance sheet management is concerned with the bank's average position in very short term funds over a span of time, money management is concerned with handling of daily fluctuations around this average. Models for these latter decisions seem to offer limited opportunity for profit improvement because of the relatively small percentage of total assets involved in short term funds.

BANKMOD I: modeling intermediate term, balance sheet management decisions

Based upon this analysis of bank management decisions, BAI has decided to concentrate on modeling the balance sheet management problem with particular emphasis on the management of the security portfolio and borrowed or purchased funds. This problem is susceptible to quantification while at the same time offering potential for a significant payout.

USE OF DISCRETE SIMULATION APPROACH

Much of the earlier work on balance sheet or asset management models was based on optimization techniques.^{5,6} The asset management problem resembled the resource allocation problem in industry and at-

tempts were made to apply linear programming techniques to its solution. With bank asset management, however, serious difficulties arose in properly quantifying the many variables and their interrelationships and in adequately specifying an objective function. Acceptance was further hindered by difficulties in gaining management understanding of the optimization approach, its benefits and its limitations. Additionally, decision-makers had fears of becoming obsolete and being replaced by some mechanized decision process. It is not surprising, therefore, that asset management optimization models have not been widely accepted. The effort expended in their development has, however, been useful in contributing to a better understanding of the problem.

The introduction of time-sharing with conversational capabilities permits use of interactive simulation to handle problems not readily amenable to optimization techniques. This approach eliminates the need to express a manager's judgment in the form of a utility function to drive an optimization model. Rather, with interactive simulation, the manager is included in the solution-finding process. He can ask "what if" questions and use his judgment to evaluate the answers.

In bank balance sheet management, finding the "best" possible policy requires evaluating often conflicting components of performance. For example, an increase in profits may coincide with a deterioration in the bank's liquidity and capital positions (i.e., cause the bank to be considered less "safe" from the depositor's point of view). It is simpler to provide the banker with information on the effects of various asset management policies and use his expertise to select the appropriate solution. The banker's judgment is brought to bear upon the problem of selecting the most desirable trade-off between profit and safety.

This approach greatly simplifies the problem, reducing both development and running costs. It is also attractive to management in that the decision making process in the simulation corresponds to that used in the real world. This understanding of the decision process facilitates management acceptance of the simulation results. In contrast, bank managements have apparently been more reluctant to accept the prescriptions of an optimization model because of failure to understand how such decisions are developed and an unwillingness to have a machine appear to take over the management role.

DESCRIPTION OF BANKMOD

The previous sections have shown need for a decision aid for managing the balance sheet and outlined the

approach BAI is taking to it. This section describes the resulting model. The paper will not discuss the underlying mathematical structure which cannot be treated adequately in the space available here. The interested reader is referred to special literature on banking, especially Gray, Kenneth B., Jr., "Managing the Balance Sheet: A Mathematical Approach to Decision Making," *Journal of Bank Research*, Spring, 1970, which presents a mathematical structure related to BANKMOD.

Overview of the model

The model is a "what if . . .", or statement projection model. Basically, it is a financial reporting system computing future bank statements resulting from the impact of environmental changes and user decisions on the initial state of the bank. These financial statements and various analytical reports provide the banker with the information he needs to select the most appropriate decision strategy.

Prior to running the model, the user develops a set of environmental assumptions consisting primarily of his forecast of future interest rates and of deposit and loan levels. He also enters the initial state of the bank including balance sheet and portfolio data.

With the assumption set and the initial state of the bank, the simulation can be run without user decision. The model then computes income, takes maturities, and adjust loans and deposits to the forecasted levels. An excess or deficit of funds is handled by sale or purchase of Fed funds.

While the simulation can be run without any user decisions, normally the banker would interact with the simulation. This interaction of the user and the model is illustrated in Figure 1.

His decisions would include reinvestment of funds becoming available (rather than simple handling through Fed funds), portfolio shifts for yield and appreciation,

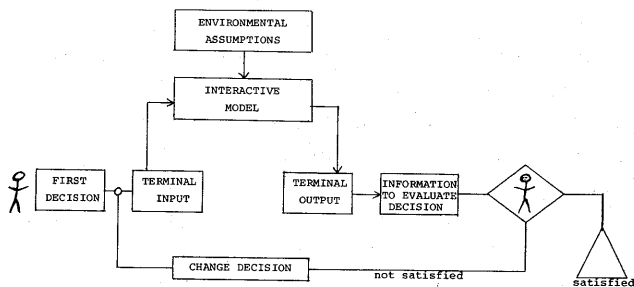


Figure 1—Interactive simulation process

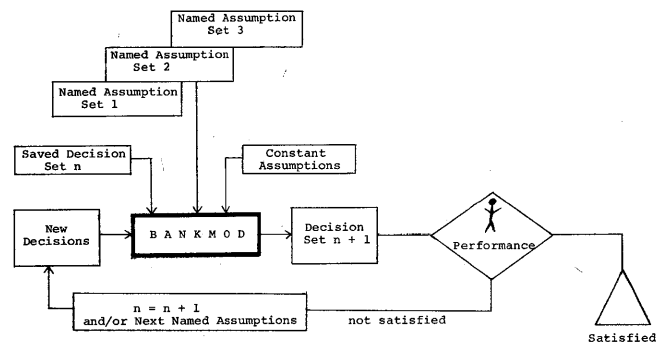


Figure 2—Iterative development of the optimal decision set

purchasing funds (e.g. issuing CDs), and changing the bank's financial leverage. Upon completion of a simulation run, the user can save the decision set and use it in a subsequent simulation where modifications to that decision set can be made. Successive iterations can be run until the decision set is considered to be reasonably optimum.

At this point another feature of the model can be utilized to test the decision set against different assumptions about the environment. Figure 2 illustrates how this feature of the model operates.

The user, after developing a satisfactory set of decisions for the most likely environment, then tests this decision set against assumption sets which represent deviations from the expected (e.g., higher loan demand and interest rates, lower loan demand and interest rates). This permits the decision set to be tested for its sensitivity to deviations of the economic environment from that which is expected. This sensitivity analysis is performed simply by calling in an alternative assumption set and running the saved decision set against it. Should analysis reveal that the decision set is not sufficiently hedged against adverse changes in the environment, the decision set can be modified appropriately.

Structuring BANKMOD into decision periods and decision points

To make BANKMOD conform as much as possible to reality and still be manageable, the planning horizon with which the banker is concerned is divided into a specified number of periods of uniform length (e.g., month, quarter). Within a period the model considers the environment to be constant. Balance sheet changes occur only at the points separating the periods (the "decision point").

In reality, however, not only does environment change continuously, but decisions are also made con-

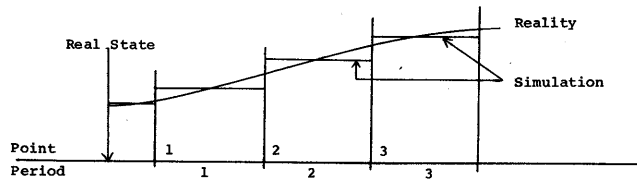


Figure 3—Continuous and discrete environment

tinuously rather than at discrete decision points. Thus the concept of a steady state during the period introduces an element of artificiality with some slight distortion of income computations. However, this distortion is not considered to be of consequence for several reasons. First of all, forecasts of deposit and loan levels are assumed to be the averages prevailing during the period. Second, although security maturities are considered to occur at the end of the period during which they actually occur, the income distortion is minor. The discrepancy in earnings is limited to the difference between the yield of the maturing security and the rate at which the proceeds of the matured security would be invested, this difference multiplied by the time between actual maturity date and the end of the period.

In any event, deviations due to the decision period convention are small compared to potential errors in the rate and level forecasts embodied in the assumptions.

Balance sheet updating

When the simulation is run all adjustments to the balance sheet can be considered as occurring at the decision points. These adjustments are of two kinds: automatic and user decisions.

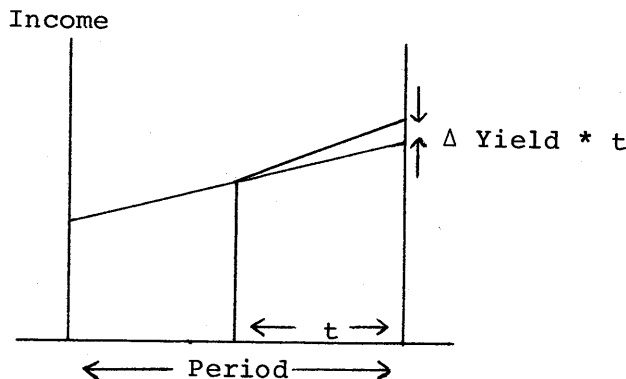


Figure 4—Effect of decisions at discrete points

Several types of automatic adjustments occur:

- Deposits and loans are adjusted to the levels forecasted in the assumption set.
- Securities, CDs, and other borrowed funds with a maturity structure are matured automatically.
- Certain balance sheet categories are adjusted in relation to others. For example, reserve is set automatically to the legal minimum required by deposits.
- Other categories are adjusted to reflect accrual of income. For example, book value of securities is adjusted to reflect amortization of premiums and accretion of discount. Equity is adjusted to reflect earnings.

These changes occur regardless of any decisions by the user.

Adjustments also occur as the result of user decisions. These include:

- Purchase or sale of securities.
- Issuance of large, negotiable CDs.
- Sales of capital notes or debentures.
- Issuance of stock.

These decisions can generate further adjustments of the automatic type. For example, a decision to issue CDs will also cause an automatic adjustment in reserve.

Because the net effect of automatic adjustments and user decisions are unlikely to be exactly offsetting, any discrepancy between a change in the source of funds and a change in the use of funds is handled by the purchase or sale of Fed funds. Excess funds earn at the Fed funds rate; a deficit is charged at the Fed funds rate.

Performance reporting

The major feedback to the user in the interactive simulation process is the system of reports available from the model. These reports resemble those which would normally be furnished by a bank's accounting system. They provide information on operating income, security gains and losses, the composition of the balance sheet, status of the portfolio and a variety of analytical ratios. One set of reports provides information regarding performance and status for the period being simulated and highlights the impact of the user's decisions for that period. These reports are designed to aid the user in further decision-making. The second group of

reports shows results for the entire simulation horizon to facilitate comparing the results of one simulation with another.

Uses of the model

The BAI simulation model is especially helpful in aiding management decision-making because it provides a consistent, comprehensive technique for examining sets of decision alternatives under a variety of assumptions about the economic environment. Specifically it aids management in analyzing a set of decisions according to the following criteria:

- Change in average yield earned on securities (paid for purchased funds).
- Related effect on liquidity and capital adequacy of portfolio changes to improve yield.
- Cost of providing funds through sale of securities if supply of funds is inadequate to meet forecasted demand.
- Opportunity to improve earnings performance by aggressively managing the security portfolio for market appreciation.
- Improvement in earnings resulting from changing the leverage of the bank (ratio of assets to equity).

The model is therefore designed to quickly summarize performance and balance sheet status according to these criteria and to evaluate a decision strategy and its sensitivity to changes in the economic environment.

SYSTEM DESIGN

Since BAI is an organization supported by a majority of banks in the U. S., it was of the highest priority to develop a model which would help not only a few very large banks but which would also be of use to medium-sized and smaller banks as well. This requirement has been recognized in several ways. A bank using BANKMOD can tailor a model to fit his bank by including only those balance sheet categories, and associated routines, which are appropriate to his bank. In addition, the model is designed to run on a nationwide time-sharing system thereby enabling any bank to access the model by simply installing a terminal and paying for the necessary CPU and connect time.

The interactive simulation process described above is embedded into a very powerful system with several large programs working together. The individual pro-

```

FM      15:53      07/02/71

FRB MEMBER (Y,N) ?Y

OPTIONAL BALANCE SHEET CATEGORIES (I FOR INCLUDE, D FOR DELETE)

ASSET
DUE FROM-FOREIGN      (1,D) ?D
MONEY MKT LOANS       (1,D) ?D
ASSET-CD'S            (1,D) ?D
ACCEPT'S&COMM'L PAPER (1,D) ?D
TREASURY BILLS        (1,D) ?I
AGENCIES               (1,D) ?I
TRADING SECURITIES    (1,D) ?D

LIABILITY
CD'S-STATE&LOCAL      (1,D) ?I
CD'S-MONEY MARKET     (1,D) ?D
EURODOLLARS-REG.D     (1,D) ?D
EURODOLLARS-REG.M     (1,D) ?D
HOLDING CO. PAPER     (1,D) ?D
CAP NOTES              (1,D) ?D

ENTER LOAN GROUPS:
NAME (ONE WORD)      ABBREVIATION (TWO LETTERS)

?INSTALLMENT         IN
?REAL-ESTATE         RE
?COMM'L-METRO        CM
?COMM'L-NATIONAL     CN
?*

ENTER DD GROUPS (EXEPT US, STATE&LOCAL):
NAME (ONE WORD)      ABBREVIATION (TWO LETTERS)

?SMALL               DS
?COMM'L-METRO        CM
?COMM'L-NATIONAL     CN
?*

LIST FORMULATION (Y,N) ?N

DECISION PERIOD (MONTH,QUARTER) ?Q

PLANNING HORIZON (NO. OF DEC.PER.) ?4

FUNDS UNIT (THOUSAND, MILLION, BILLION) ?T

ENTER PORTFOLIO-ARRAY SPECIFICATIONS :
CAT.      MIN.PURCH.YIELD  INCREMENT  NO. OF POINTS
BIL ? 2           1           6
GOV ? 2.5         .5          10
AGC ? 3           1           6
MUP ? 3.5         .5          10
MUG ? 4           .5           8

FM END
    
```

Figure 5—Formulate mode

grams or modes of the system are:

- Formulate
- Assumption
- Real State
- Input-Forms
- Query
- Simulate

Originally, these modes are programs in the *shared* library of the T/S company. When run, they build and utilize files in the user's *private* library. This guarantees security for the user's proprietary data, such as the

initial state of the bank and forecasts of loan and deposit levels. The major features of the model can be described in terms of these modes.

Formulate Mode

In the Formulate Mode (FMODE) the user interactively tailors the model to fit his needs. Figure 5 illustrates how the user formulates a specific model for his bank. The pattern of interaction with the user is for the program to print the data to the left of the question mark while the user's response immediately follows the question mark.

The user establishes the various parameters of the model. In addition to the basic balance-sheet structure, he can add optional asset and liability categories which may be appropriate to his bank. He can also provide further breakdowns of loan and deposit categories as desired.

In this mode the user also defines the length of a period to fit his requirements for accuracy, and the number of periods to be simulated. He also defines the size and degree of resolution incorporated in the portfolio array structure. FMODE uses this information to specially tailor all the other modes. That is, when the other modes are run, all their arrays, records and files are custom-built in order to run the program more efficiently, particularly to minimize response time.

The user normally runs FMODE once to "establish" his model bank, but may set up additional models with different period lengths and different levels of balance sheet detail.

Assumption Mode

In the Assumption Mode (AMODE) the user describes the economic and institutional environment in which the simulation will take place. Certain assumptions, primarily economic, must be forecast for each period in the simulation. Examples are yield curves, loan and deposit levels and Regulation Q limits. (See Figure 6 for an illustration of a completed page of the named assumption input form.) Other assumptions are considered to remain constant over the entire planning horizon. These include assumptions such as bid-ask spreads and tax rates.

The user can establish several sets of economic assumptions and assign each a name. One set may represent the environment considered most likely to prevail. Another may be based upon a more active economy with greater loan demand and higher interest rates. A third may forecast a less active economy with all its

		INPUT FORMS NAMED ASSUMPTI			
ASSUMPTION SET NAME		<u>EXPECTED</u>			
LINE					
KEY MARKET DETERMINED RATES (XX.X)					
115	DAILY FUNDS	<u>3.0</u>	<u>3.5</u>	<u>4.0</u>	<u>4.0</u>
GOVERNMENTS-YIELDS TO MATURITY (BASED ON ASKED PRICES)					
117	90 DAYS	<u>3.3</u>	<u>3.3</u>	<u>3.5</u>	<u>3.5</u>
118	180 DAYS	<u>3.4</u>	<u>3.7</u>	<u>3.7</u>	<u>3.7</u>
119	30 YEARS	<u>6.0</u>	<u>5.7</u>	<u>5.7</u>	<u>5.7</u>
IF A HUMPED CURVE COMPLETE THE FOLLOWING TWO LINES: (IF NON-HUMPED ENTER ZEROES)					
121	MATURITY OF MAX YIELD(YRS)	<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>
122	MAXIMUM YIELD	<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>
AGENCIES-YIELDS TO MATURITY (ASK PRICES)					
125	1 YEAR	<u>3.7</u>	<u>3.7</u>	<u>3.9</u>	<u>3.9</u>
126	2 YEARS	<u>4.1</u>	<u>4.0</u>	<u>4.0</u>	<u>4.0</u>
127	15 YEARS	<u>6.6</u>	<u>6.4</u>	<u>6.0</u>	<u>6.0</u>
MUNICIPALS-PRIME YIELDS TO MATURITY (ASK PRICES)					
130	1 YEAR	<u>2.7</u>	<u>2.7</u>	<u>2.7</u>	<u>2.7</u>
131	2 YEARS	<u>3.0</u>	<u>3.0</u>	<u>3.0</u>	<u>3.0</u>
132	30 YEARS	<u>5.7</u>	<u>5.7</u>	<u>5.5</u>	<u>5.5</u>
PRICE OF BANK'S STOCK					

Figure 6—Illustration of assumption input

implications. Each set, however, must be internally consistent. These sets of assumptions are used to check the sensitivity of one set of decisions against different economic forecasts.

The amount of data required for each named assumption set are approximately 100 items per period in the simulation horizon. Additionally, about 75 items are required to specify certain institutional factors which remain constant over all periods in the simulation and all environment forecasts. The actual number of items will depend on the size of the formulated model.

Because of the amount and importance of the data entered as assumptions, considerable attention was given to insure the efficiency and accuracy of the input procedures. The model will generate forms for the banker to use in assembling the required data. Once the data have been prepared, a technical assistant can be

used to enter the data through the terminal keyboard. As the data are entered, the program intensively checks for technical validity and, to some extent, checks for logical validity as well.

Another feature permits the user to develop a set of assumptions which are a partial modification of an established set. The established set is copied under a new name and the user then makes the desired modifications to this newly generated set.

Real State Mode

The Real State Mode (RMODE) is used to enter the most recent actual "state of the bank" data. This includes the current balance sheet amounts and a somewhat aggregated description of the portfolios.

PAGE 1

REAL STATE INPUT FORMS	
DATE 5/4/71	
ASSET	BOOK VALUE (THOUSANDS OF DOLLARS)
CASH	CSH 450
RESERVE	RES 2100
DUE FROM	DUE 500
ITEMS IN PROC. COLL.	PRC 750
DAILY FUNDS SOLD	DFS 0
TREASURY BILLS	BIL 2382
GOV'TS	GOV 5985
AGENCIES	AGC 0
MUNICIPALS-PRIME	MUP 5400
MUNICIPALS-GOOD	MUG 0
INST. LOANS	LIN 3681
RE LOANS	LRE 5622
LOANS-COMM'L METRO	LCM 4254
LOANS-COMM'L NAT'L	LCN 3113
BLDGS&EQUIP	B&E 668
OTHER ASSETS	OTA 600

Figure 7—Illustration of real state input

The model will generate forms for assembling the necessary data. (See Figure 7 for an illustration of a completed page of the real state input form.) The data would then be entered manually through the terminal keyboard. For those banks which have automated general ledger and portfolio files, it would be possible to use a utility program to transform the data to BANKMOD specifications and transmit it directly into the model.

Input Forms Mode

This mode (IMODE) provides the input forms described for the AMODE and RMODE. The user specifies the number of copies and whether they should be printed in-house on the terminal or on a high speed printer at the computer center for mailing to the user.

Query Mode

The Query Mode (QMODE) provides for convenient access to certain permanent files that are kept on the computer. For example, it might be necessary to get the contents of the portfolio files if this information has been misplaced since the last run, or it might be of interest to check the nature of certain decisions for a certain period while performing a sensitivity analysis.

Simulate Mode

The Simulate Mode (SMODE) is the heart of the whole BANKMOD system. Here all the work, outlined above in the description of BANKMOD, is carried out. The user proceeds period by period from the initial state of the bank (as entered in RMODE) to the end of the planning horizon. In doing so, the user requests specific information, makes decisions regarding the discretionary assets and liabilities, and calls for various performance and analytical reports.

Within each period, the decision-making process is an interactive conversation as portrayed in Figure 1 above. The user requests information about the status of the bank, makes appropriate decisions, and calls for reports of various levels of detail to evaluate the impact of his decisions. These reports are called "snapshot" reports and show performance data for only the period being simulated. When satisfied with the decisions for a period, the user causes the simulation to proceed to the next period until the end of the simulation horizon has been reached.

Once the user has completed the decision-making process over the entire horizon, he can obtain so-called "horizon" reports summarizing performance and status over each period and in total for all periods. These reports can be used for comparing the results of separate simulation runs. This can be done to evaluate different decision sets (representing different strategies) or for evaluating a decision set against several different environments.

The interactive decision process is discussed in more detail in the following section of the paper where it is used to illustrate how the model handles the man-machine interface.

COMMUNICATIONS HANDLING AND THE MAN-MACHINE INTERFACE

As has been intimated, it was of the highest priority to develop a model which would help not only a few giant banks but which would be of use to medium-sized banks and possibly small banks as well. This directly implies that the model must be designed to operate in a way understandable to the bank management—its use must not depend on a sophisticated management science department since only the largest banks have such departments. Further, the user should be able to run the model and interpret the results with a minimum of formal instruction. It seems clear, therefore, that the acceptance and the ultimate value of the model depended heavily on the quality of its interactive capabilities. That is to say, successful use of the model depends on a satisfactory solution of the man-machine interface problem.

It has been frequently pointed out that adequate handling of the man-machine interface is one of the most difficult and vexing problems in using computer time-sharing systems (or for that matter, on-line real-time systems as well).⁸ The difficulty involves achieving accurate, unambiguous and efficient communication between the user and the computer. The user must find it easy and natural to enter input data, to guide the simulation process, and to extract and interpret simulation outputs. The next sections describe how BANKMOD satisfies these requirements and concludes with an illustrative run of the Simulate Mode.

Input

BANKMOD offers the user format-free input with extensive error checking. The program prevents the user from causing a program or system breakdown by

accidentally entering faulty responses. It also prevents the user from entering information which does not meet the built-in logic tests.

User-computer interaction

Great effort has been spent to make user-computer interaction as easy and natural as possible. First, the user communicates with the model using terms familiar to him. Second, the communication process is standardized throughout the model, so that whatever mode is being used, the user employs the same technique for entering data, issuing commands, and requesting output.

In addition, the method of communication with the model has been reduced to two types: indicated response and sequence-free commands. The indicated response is used where the input cannot be sequence free, such as entering certain parameters at the initiation of the run. A simple command structure is used to enter user decisions, requests for information and reports, and simulation control commands on a completely sequence-free basis.

Once the user has become familiar with these communications standards, he can run the simulation with a minimum of distraction for system mechanics and concentrate on the problem-oriented aspects of the simulation.

Output

The speed and format of the output are of critical importance. Speed of response is facilitated by the design of the system. Most of the output requires only nominal calculation with very short response time. Major computations with longer delays (of the order of 10 to 20 seconds) are made only when moving from one simulation period to the next. This design provides the user with quick response in decision-making interaction and makes the longer response time when the user is psychologically better conditioned to endure the delay.

The proper selection and presentation of output information is also of critical importance for a model such as this. Not only is the total amount of data quite large, but decisions must be based upon this data. One of the major research efforts was, therefore, devoted to structuring the output to maximize the information conveyed and minimize the total data presented. For the first evaluation of the effect of a decision, a banker can ask for a report of a few key indicator variables.

ASSUMPTION CATALOGUE (Y,N) ?Y
 NAMED ASSUMPTIONS :
 EXPECTED
 TIGHT
 EASY
 ASSUMPTION NAME ?EXPECTED
 SAVED DECISIONS (Y,N)?Y
 DECISION CATALOGUE (Y,N) ?Y
 NAMED DECISIONS :
 WPH-TEST1
 WPH-TEST2
 DECISION NAME ?WPH-TEST2

*** DECISIONPOINT 0: 05/04/71 ENTER COMMANDS ***
 ?*N
 ASSUMPTION: EXPECTED
 DECISION : WPH-TEST2

*** DECISIONPOINT 1: 07/01/71 ENTER COMMANDS ***
 ?*FUN
 FUNDSBAL.: 782.693
 ?*D MUP

MAT. POINT	AMT PAR	PURCH. YIELD	COUPON
?10	1000		
?*D GOV			

MAT. POINT	AMT PAR	PURCH. YIELD	COUPON
?10	-200	3	
?*FUN			

FUNDSBAL.: -88.6435

?*FLASH

ASSUMPTIONS : EXPECTED SAVED DECISIONS : WPH-TEST2

	BEFORE DECISIONS	AFTER DECISIONS	CHANGE	PERCENTAGE CHANGE						
				-15 %	-10 %	-5 %	0 %	5 %	10 %	15 %
NOI/SHARE	1.51	1.81	.29							+++++
NI/SHARE	1.51	-1.32	-2.83							-----
LIQUIDITY RATIO	95.6%	93.2%	-2.4%							XXXX
CAPITAL RATIO	23.8%	21.7%	-2.1%							XXXXXXXXX

?*I BIL

PAR= 1410 BOOK= 1398.43

PORTFOLIO SUMMARY (Y,N) ?Y

MATURITY POINT	2	PURCH. 3	YIELD 4	5	6	7
2	0	0	750	0	0	0
3	0	0	660	0	0	0
4	0	0	0	0	0	0
5	0	0	0	0	0	0

?*END

SAVE DECISIONS (Y,N) ?Y
 DECISION CATALOGUE (Y,N) ?N
 NAME ?WPH/TEST3
 WPH/TEST3 SAVED ON DF3

SM END SIGN OFF
 READY

Figure 8—Illustration of a simulate mode run

If necessary or desired he can then request additional reports providing greater levels of detail. Without this report structure whereby the user requests a specific type of information and specifies the level of detail, the overwhelming amount of information would make the interactive approach totally impractical.

Sample run

To illustrate the design concepts described above, this section concludes with a sample run. Since the communications standards are uniform throughout the model and the Simulation Mode is the most extensively used, a run of this mode is used for illustrative purposes. (The sample run referred to in the material following is reproduced in Figure 8.)

After the user has started SMODE, the computer is in one of two states: either awaiting a pre-defined indicated response or awaiting a command. Examples of pre-defined indicated responses are shown in Figure 8 at the start of the SMODE run. The program here requests information regarding the assumption set and decision set to be used in the simulation. Any input other than what is indicated will be rejected resulting in a repetition of the inquiry. The request and response are kept as short as possible.

Commands are given in those parts of the run which are sequence-free. Commands are essentially of four types—Control, Information, Decision and Report—and all commands begin with the character “*”.

Control commands steer the flow of the program. An example in Figure 8 is the command “*N” which causes the simulation to move from decision point 0 (the starting point) to decision point 1 (the beginning of the first full calendar quarter). Another example near the bottom of Figure 8 is “*END” causing the simulation to proceed to the last period and terminate. Another control command, not illustrated in the example is “*PAUSE”. This command stores the status of the simulation permitting the user to sign off and later resume the simulation at the point where he paused. This has proved to be very convenient in the use of the model since the user can interrupt the simulation to consider decision strategy or avoid excess fatigue without elaborate requirements for saving the status of the simulation.

The information command is a multi-purpose command of the format “*I cat”. The abbreviation “cat” signifies that the user specifies the balance sheet category for which he desires information. In the sample run, the command “*I BIL” shows the use of this command to obtain information about the bank’s

holdings of Treasury Bills. After providing the basic information, the program asks the user if he wishes detailed portfolio data. If the user provides the response “Y”, the portfolio array as shown in the lower portion of Figure 8 is printed.

The decision command is also a multi-purpose command taking the format “*D cat” where “cat” refers to the balance sheet category to be changed. The use of this command is shown at the mid-point of Figure 8. The command “*D MUP” indicates that the user wants to enter a decision to buy or sell prime municipals. The computer responds with the format readings prescribing how the data is to be entered. The command “*D GOV” illustrates a command to buy or sell U. S. Government bonds.

The run illustrates several uses of the report command. The command “*FUN” illustrates a request for the current funds balance (net funds currently available for investment). This is essentially a one-item report. A more elaborate report is obtained by the command “*FLASH” which shows the effect of the decisions on certain key indicators of performance. The report combines tabular presentation of data with a graphical display of the direction and magnitude of the changes resulting from the decisions. Additional “snapshot” reports are available providing greater levels of detail of performance and bank status for the period being simulated. Also, at the completion of the simulation, the user can request various “horizon” reports which summarize the simulation over all the periods in the simulation.

The sample run terminates after asking the user whether or not the decision set should be named and saved for future simulations.

REFERENCES

- 1 BOOZ-ALLEN & HAMILTON INC
The challenge ahead for banking: a study of the commercial banking system in 1980
Chicago Illinois August 1970
- 2 C F HAYWOOD L R MCGEE
The expansion of bank funds in the nineteen-seventies
Association of Reserve City Bankers Chicago 1969
- 3 G C FISCHER Ed
Commercial banking 1975 and 1980
Robert Morris Associates Philadelphia 1970
- 4 R I ROBINSON R H PETTWAY
Policies for optimum bank capital
Association of Reserve City Bankers Chicago 1967
- 5 K J COHEN F S HAMMER
Analytical methods in banking
Richard D Irwin Homewood Illinois 1966

6 K J COHEN F S HAMMER

Linear programming and optimal bank asset management decisions

Journal of Finance XXII May 1967

7 J B BOULDEN

Instant modeling in Corporate simulation models

A N Schrieber Ed Seattle Washington University of Washington 1970

8 H SACKMAN

Computers, systems science and evolving society

John Wiley and Sons Inc New York 1967

9 K B GRAY JR

Managing the balance sheet: a mathematical approach to decision making

Journal of Bank Research Spring 1970

Simulation of large asynchronous logic circuits using an ambiguous gate model

by S. G. CHAPPELL

Bell Telephone Laboratories
Naperville, Illinois

and

S. S. YAU

Northwestern University
Evanston, Illinois

INTRODUCTION

Digital logic simulation is the process whereby the action of a logic circuit due to a specified input is predicted based upon some model of the circuit. Logic simulation is becoming increasingly necessary as larger and more complex computers are built. Because of the cost of building hardware it is not wise to commit a circuit design to manufacture without first verifying the operation of the circuit by simulation. This is true even for large computers (say 50,000 gates) where simulation will eliminate many logic errors and may save construction of a prototype model. Simulation may be used to predict the output of the circuit due to specified faults as well as to predict the output of the good (fault free) circuit. A dictionary is generally compiled of the output of the circuit in the presence of known faults. By comparing the actual (perhaps faulty) circuit output to the correct output, it is possible to detect and diagnose a fault in the circuit.

Many simulators have been proposed¹⁻⁹ which allow simulation of the good circuit and several have been proposed which allow simulation of the traditional stuck-at (stuck-at-one and stuck-at-zero) faults. Among these simulators, the following are more significant. Eichelberger¹ first proposed a simulator for logic circuits, based on the work of Yoeli and Rinon² which uses a ternary logic 0, $\frac{1}{2}$ and 1 where logical 0 and 1 are the Boolean 0 and 1 and the $\frac{1}{2}$ represents a don't-know value. However, his simulation and hazard detection are based on the Huffman Model which is not an accurate representation of a general asynchronous logic circuit because the delay in all the gates is lumped into the delay elements. Seshu's Sequential Analyzer³ for

logic circuits provides fault simulation and race analysis which is also based on the Huffman Model. However, only binary simulation logic is used and even combinational logic is not simulated correctly since hazards are suppressed due to the use of the leveling technique. Leveling means the output of a gate is not calculated until all its inputs are known. In this way the output of each gate is calculated only once. Later Chang⁴ extended the simulator to include shorted input diodes on a DTL gate. Szygenda, Rouse and Thompson⁵ proposed a simulator which uses ternary simulation logic and provides various gate delays and ambiguity regions (regions where the simulation model cannot predict the output of a gate). However, the third logic value is only used for circuit initialization. During simulation a Potential Error Flag is used to represent the ambiguity region and an additional logic element is required to manipulate this flag. In addition, the uncertainty region associated with the turn-on delay is assumed to be the same as that for turn-off delay for a gate.

The simulator proposed in this paper is the only simulator which is able to accurately simulate the effects of shorted diode and shorted net (gate outputs) faults in an asynchronous sequential circuit. In addition, the gate model used here allows specification of minimum and maximum turn-on and turn-off delays where the interval between the minimum and maximum delay is treated as a third simulation value x (the ambiguity or don't-know value). The ambiguity value allows efficient handling of each gate (no extra elements are required) and the use of a new high-frequency rejection technique. This technique provides easy suppression of transient input conditions of shorter dura-

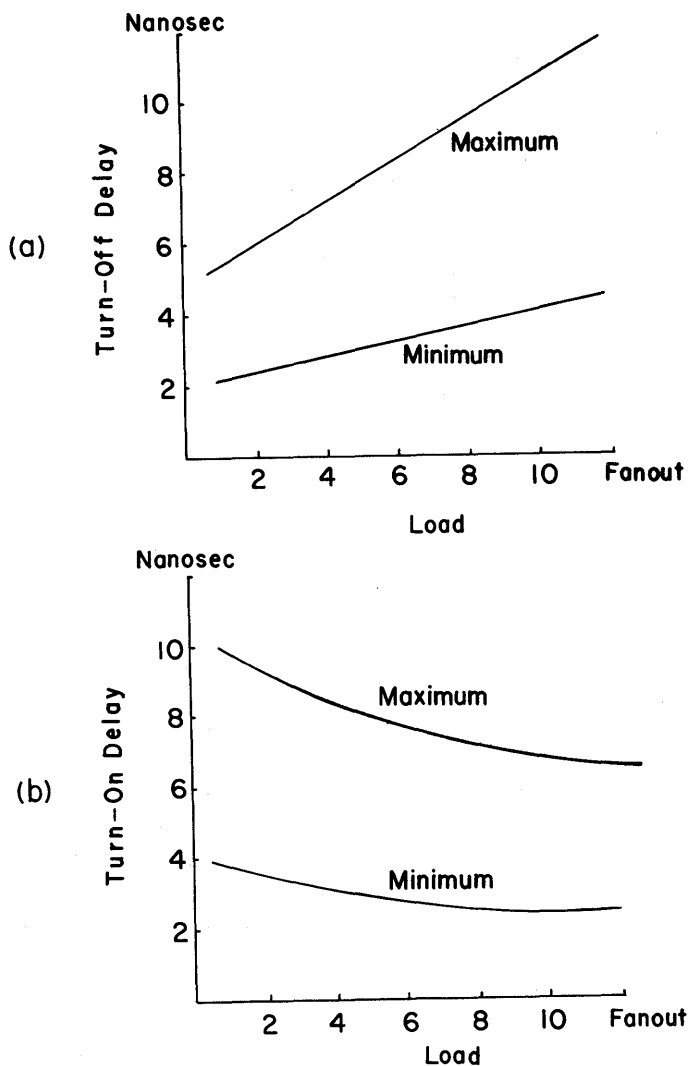


Figure 1—Load vs transition time curves for TTL logic

tion than the appropriate minimum transition delay. The gate model used allows detection of certain constrained hazards and produces a worst case timing analysis of the circuit (based on the transition delays assigned to each gate) for both the good and the faulty circuit. This simulator has been implemented for circuits of up to 50,000 gates, and its speed is comparable to that of the latest simulator.⁵

THE GATE MODEL

In order to provide an accurate simulation of a logic circuit, an accurate model of each logic element is necessary. Only gates will be simulated here since the actual logic circuit is composed only of interconnected gates. Attempts to simulate larger modules such as

flip-flops and registers may introduce logic and timing errors. The only exception to this rule is that a pure delay element is allowed to simulate actual delay lines or long wiring runs.

If the load versus time curve for a typical TTL gate shown in Figure 1 is examined, it is apparent that the turn-on and turn-off (transition) delays are different. It is also known that there is considerable variation in transition delays among supposedly identical gates. In addition, such factors as the loading, length of the output net and temperature, affect the speed of a gate. However, if reasonable design constraints are imposed and the gates are reliably manufactured, the transition delays will usually fall within certain bounds. Therefore, a logic gate may reasonably be characterized by its minimum and maximum turn-on (0 to 1 transition) and turn-off (1 to 0 transition) delays.

The turn-on (turn-off) delay is the time between application of the input signal and the time the output signal reaches 90 percent (10 percent) of its final value (initial value). If the gate is operating in saturation, the turn-off time includes the storage time as well as the decay time of the gate (transistor). However, this characterization of a gate is very general and may be applied to any gate regardless of its mode of operation or location in the logic circuit. Another factor which must be considered is the high-frequency rejection of a logic gate. That is, a gate will not respond to an input pulse of shorter duration than the appropriate minimum transition delay. Therefore, the gate model must perform both high-frequency rejection and account for variations in gate transition delays. In this paper, the gate model will be referred to as the gate, while the real gate will be called the real or actual gate.

Assume the gate G may be characterized by the following four parameters:

- a = minimum turn-on delay
- b = maximum turn-on delay
- c = minimum turn-off delay
- d = maximum turn-off delay

The transition delays for G may be set to any integral values and are typically selected based on statistical information about the behavior of the gate being used for some given load. If $a=b$ and $c=d$, then the gate model G is said to be *unambiguous*. If $a < b$ and $c < d$, then the time $(b-a)$ or $(d-c)$ is the *ambiguity region* for the gate G , and the model is said to be *ambiguous*.

Let $*$ denote the absence of an element from a vector. Define the three operators $L\{G\}$, $C\{G\}$ and $R\{G\}$ on a 3-element vector $G = (X, Y, Z)$ as $L\{G\} = X$, $C\{G\} = Y$ and $R\{G\} = Z$. Similarly for a 2-element vector, $G = (X, Z)$, define $L\{G\} = X$ and $R\{G\} = Z$.

The objectives of the gate model are: 1) to generate and propagate an ambiguity region which represents the uncertainty about the exact behavior of the gate, and 2) to perform high-frequency rejection which means the gate output does not respond to transient input conditions of duration shorter than the appropriate minimum transition delay.

The above objectives are achieved by using G_I , G_I , G_D and G_F . G_I , called the *instantaneous output* of the gate G , is based only on the truth table for G . $G_I = (q_1, q_2)$, where $q_1 = 0, 1$ or x and $q_2 = 0, 1$ or $*$, is called the *modified instantaneous output vector*. G_I is used to introduce an ambiguity region into the output transition of the gate G (if necessary) when the input transition was nonambiguous. $G_D = (q_3, q_4, q_5)$, where $q_3 = 0, 1$ or x , $q_4 = 0, 1, x$ or $*$ and $q_5 = 0, 1, x$ or $*$, is called the *filtered output vector*. G_D is used to perform the high-frequency rejection and is calculated from G_I and the *present free output vector* \tilde{G}_F of the gate. The *next free output vector* $G_F = ([q_3, t_1], [q_4, t_2], [q_5, t_3])$ is obtained from G_D by associating the transition times t_1, t_2 and t_3 with the appropriate elements of G_D . The transition times t_1, t_2 and t_3 are the times when the output of G assumes the values q_3, q_4 and q_5 respectively. It is noted that no times are associated with the $*$ elements.

For the vector \tilde{G}_F , q_3 is the present value assigned to the output of G and t_1 is the time G assumed the output value q_3 . The values q_4 and q_5 are *future values*, which may become the output of G at times t_2 and t_3 respectively. The values q_4 and q_5 will not become the output of G if they are the result of a transient input condition such that the high-frequency rejection will remove q_4 and q_5 from \tilde{G}_F as explained later.

Each gate G always has a \tilde{G}_F associated with it to represent its free output values. The remaining vectors G_I and G_D as well as G_I are used only to calculate the next G_F which will become the present free output vector \tilde{G}_F for future calculation. The actual or *constrained output* of a gate G is the output of G in the presence of any fault being simulated. Only one circuit (i.e., the good circuit or one faulty circuit) is simulated during any simulation run.

To illustrate the procedure, consider the NAND gate G with two inputs i and j shown in Figure 2. Let G have a minimum and maximum turn-on delay of 4 and 6 nanoseconds respectively and a minimum and maximum turn-off delay of 4 and 5 nanoseconds respectively. Assume that the inputs in Figure 2(a) are $i=0$ and $j=1$ and that they have not changed for at least 6 nanoseconds so that the output of G is 1. Also assume that the output of G changed to 1 at time t_1 and that i changes from 0 to 1 at t_0 . As shown in Figure 2(b), G must change from 1 to x at t_0+4 and change from x to 0 at t_0+5 , provided that no subsequent input

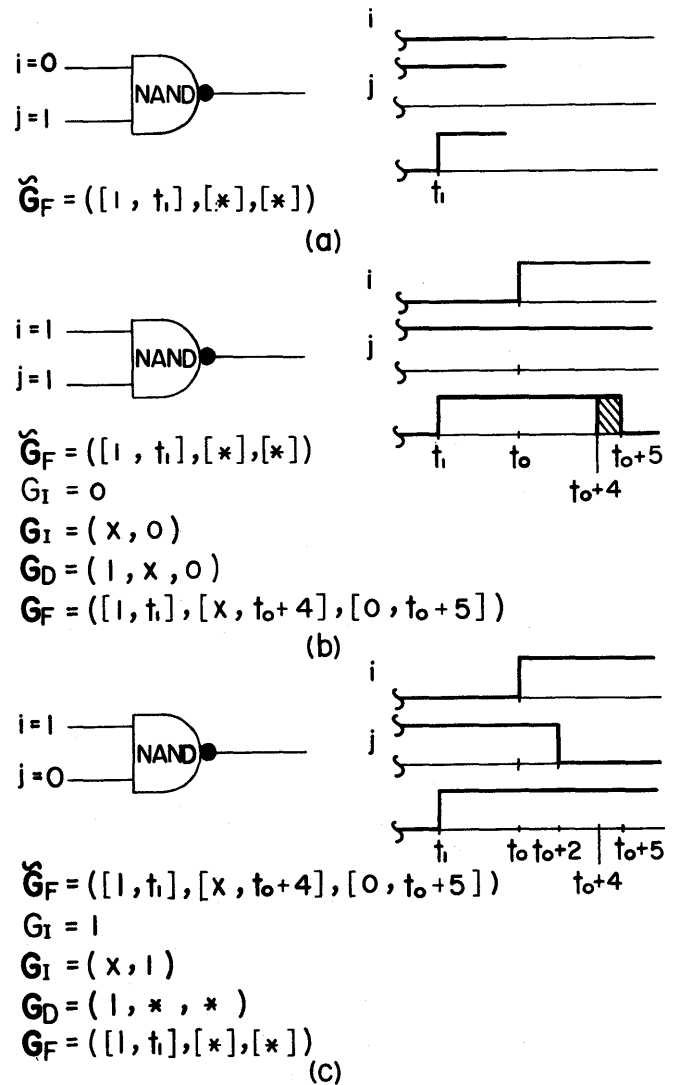


Figure 2—Gate response to a transient input condition

transitions occur. The x and 0 are called *future values* since they may become the output of G later. However, assume j also changes from 1 to 0 at time t_0+2 . Then since the future values have not yet been assigned to the gate, the input conditions which would have caused the 0 input were not of sufficient duration to allow the output transition to occur. Therefore, the gate G cannot respond to the transient input conditions and the x and 0 are removed from the list of future values on gate G .

To calculate the next free output vector G_F as the result of an input transition, the first step is to find the instantaneous output value G_I according to the truth table for the gate. G_I is not necessarily the actual output of the gate. For the example shown in Figure 2(b), $G_I=0$. The truth tables for a 2-input NAND and a 2-input OR gate are shown in Figure 3.

	O	I	X
O	I	I	I
I	I	O	X
X	I	X	X

NAND

	O	I	X
O	O	I	X
I	I	I	I
X	X	I	X

OR

Figure 3—Two-input truth tables

The second step is to find the modified instantaneous output vector G_I from G_I . Let $k=1$ or 0 and let k' be the complement of k . If either G_I or $R\{\tilde{G}_F\}$ is the value x , then $G_I=(G_I, *)$. If $G_I=k$, $R\{\tilde{G}_F\}=k'$ and the gate G is ambiguous, then $G_I=(x, G_I)$. This inserts the ambiguity region on the output of gate G when the input transition is nonambiguous but the gate is ambiguous. If $G_I=R\{\tilde{G}_F\}$ then no further processing occurs (\tilde{G}_F is left unchanged) since the output of the gate will not change. Because of this selectivity no two adjacent elements of G_D or G_F can have the same value.

The third step performs the high-frequency rejection, based on the modified instantaneous output vector G_I and the present free output vector \tilde{G}_F , to produce the filtered output vector G_D . To calculate G_D the elements of G_I are considered one at a time (first $L\{G_I\}$ then $R\{G_I\}$). Let the value h , where $h=0, 1$ or x , be the element of G_I under consideration. The high-frequency rejection is best explained by considering three cases.

Case I. If $\tilde{G}_F = ([i, t_1], [*], [*])$, where $i=0, 1$ or x , then $G_D=(i, h, *)$. This is obvious since it simply says that at some time in the future, the output value of G will change from i to h . There are no restrictions on this transition.

Case II. If $\tilde{G}_F = ([i, t_1], [j, t_2], [*])$, where $i, j=0, 1$ or x , then consider the three groups of calculations listed in Table I. All the Group 1 calculations are of the form $\tilde{G}_F = ([h, t_1], [j, t_2], [*])$, where $j \neq h$, and the result is $G_D=(h, *, *)$. This means that previous inputs to gate G caused an output j to be calculated. However, since the value j is not the present value on G (the present value is $L\{\tilde{G}_F\}$), the input which caused the output j was not of sufficient duration to allow G to make the transition from h to j . The present inputs to G produce the modified instantaneous output element

h which is exactly the present output of G . Therefore, the effect of the transient input condition is suppressed (high-frequency rejection) and the output of G remains unchanged.

The Group 2 calculations are similar to the Group 1 calculations in that the output due to a transient input condition is suppressed. The Group 2 calculations are of the form $\tilde{G}_F = ([i, t_1], [j, t_2], [*])$, where $i=0, 1$ or x and $j=0$ or $1, i \neq h$, and $j \neq h$. The result is $G_D=(i, h, *)$. The transient input condition which caused the output j is suppressed, but since $i \neq h, h$ must be added to G_D .

The Group 3 calculations involve no high-frequency rejection since they are simply transitions from h to h' (where h' is the complement of h and we define $x'=x, 0'=1$ and $1'=0$). Therefore, $\tilde{G}_F = ([h, t_1], [x, t_2], [*])$ produces $G_D=(h, x, h')$. No output suppression is performed because x is either a 0 or 1 . Then, on the actual gate only an h to h' transition will occur sometime between the h to x transition time t_1 and the x to h' transition time t_3 .

Case III. If $\tilde{G}_F = ([i, t_1], [x, t_2], [i', t_3])$, for $i=0$ or 1 , then consider the following two types of h . The first type is $h=i$. Since $x=i$ or $x=i'$, the present \tilde{G}_F may be reduced to $\tilde{G}_F = ([i, t_1], [i', t_3], [*])$. Then, from Group 1 of Case II above $G_D=(i, *, *)$. The second type is $h=x$. Since all of the transitions which have been considered are to occur at some time in the future, and since any gate may have any time parameters, it is of no consequence how far in the future the transitions will occur. Therefore, rather than considering $\tilde{G}_F = ([i, t_1], [x, t_2], [i', t_3])$, only a partial \tilde{G}_F

TABLE I—Calculation of G_D Based on \tilde{G}_F and h

\tilde{G}_F	h	G_D
Group 1		
$([0, t_1], [1, t_2], [*])$	0	$(0, *, *)$
$([0, t_1], [x, t_2], [*])$	0	$(0, *, *)$
$([1, t_1], [0, t_2], [*])$	1	$(1, *, *)$
$([1, t_1], [x, t_2], [*])$	1	$(1, *, *)$
$([x, t_1], [0, t_2], [*])$	x	$(x, *, *)$
$([x, t_1], [1, t_2], [*])$	x	$(x, *, *)$
Group 2		
$([0, t_1], [1, t_2], [*])$	x	$(0, x, *)$
$([1, t_1], [0, t_2], [*])$	x	$(1, x, *)$
$([x, t_1], [0, t_2], [*])$	1	$(x, 1, *)$
$([x, t_1], [1, t_2], [*])$	0	$(x, 0, *)$
Group 3		
$([0, t_1], [x, t_2], [*])$	1	$(0, x, 1)$
$([1, t_1], [x, t_2], [*])$	0	$(1, x, 0)$

needs to be considered; i.e., $\tilde{\mathbf{G}}_F = ([x, t_2], [i', t_3], [*])$. Then, again from Group 1 of Case II above, we have the partial $\mathbf{G}_D = (x, *, *)$ and the total $\mathbf{G}_D = (i, x, *)$. In summary $\tilde{\mathbf{G}}_F = ([i, t_1], [x, t_2], [i', t_3])$ may produce $\mathbf{G}_D = (i, *, *)$ or $\mathbf{G}_D = (i, x, *)$.

The fourth step simply associates the proper time with the elements of \mathbf{G}_D to produce the next \mathbf{G}_F . Let $\tilde{\mathbf{G}}_F = ([h, t_h], [m, t_m], [n, t_n])$, where $h=0, 1$ or x , m and $n=0, 1$ or x or $*$, and t_h, t_m and t_n are the transition times associated with the output values. Similarly, let $\mathbf{G}_D = (i, j, k)$ where $i=0, 1$, or x , and $j, k=0, 1$ or x or $*$. By definition $i=h$ since $L\{\tilde{\mathbf{G}}_F\}$ = the present output of the gate. If $j=m$ ($k=n$), then t_m (t_n) remains unchanged. Otherwise, $m \leftarrow j$ ($k \leftarrow n$) and $t_m \leftarrow t_j$ ($t_n \leftarrow t_k$), where the left arrowhead means "is replaced by," and t_j (t_k) is the transition time associated with the transition to the value j (k). The transition times t_j and t_k are determined from h and the gate timing parameters as shown in Table II. The transition times t_j and t_k tell when the output transition will occur if it is not suppressed later.

The last step is to rename the next free output vector \mathbf{G}_F to the present free output vector $\tilde{\mathbf{G}}_F$. This is simply a step in the calculations and does not represent a change in the output value of the gate G because $L\{\tilde{\mathbf{G}}_F\} = L\{\mathbf{G}_F\}$. This step completes the calculation of $\tilde{\mathbf{G}}_F$ from the selected element of \mathbf{G}_I . The entire process is repeated for the next element of \mathbf{G}_I if \mathbf{G}_I contains no $*$ element.

The free output vector $\tilde{\mathbf{G}}_F$ of an isolated gate G may be altered by input transitions as explained above. The only other operation performed on $\tilde{\mathbf{G}}_F$, called the *assignment operation*, occurs when a future output value is actually assigned to become the new output of G . Let \mathbf{G}_F be the next free output vector and $\tilde{\mathbf{G}}_F$ be the present free output vector of G . Then $L\{\mathbf{G}_F\} \leftarrow C\{\tilde{\mathbf{G}}_F\}$, $C\{\mathbf{G}_F\} \leftarrow R\{\tilde{\mathbf{G}}_F\}$ and $R\{\mathbf{G}_F\} \leftarrow *$. $C\{\tilde{\mathbf{G}}_F\}$ cannot be $*$ since this operation cannot occur unless the time t_n associated with $C\{\tilde{\mathbf{G}}_F\}$ is equal to t_0 the present time.

It can easily be seen that the described operations on $\tilde{\mathbf{G}}_F$ are closed. Any \mathbf{G}_F containing only one non- $*$ element is allowable. Also from Table I, any \mathbf{G}_F of the form $([i, t_1], [x, t_2], [i', t_3])$ is allowable for $i=1$ or 0 .

TABLE II—Transition Times for Various Output Changes

$j(k)$	$m(n)$	$t_j(t_k)$
0	x	$a+t_0$
x	1	$b+t_0$
1	x	$c+t_0$
x	0	$d+t_0$

Since $\mathbf{G}_F = ([i, t_1], [*, *])$ is allowable for any $i=0, 1$ or x , then an assignment on any $\tilde{\mathbf{G}}_F = ([i, t_1], [j, t_2], [*, *])$ for $i, j=0, 1$ or x , would produce an allowable \mathbf{G}_F . Similarly, any assignment on $\tilde{\mathbf{G}}_F = ([i, t_1], [x, t_2], [i', t_3])$ for $i=1, 0$, would result in $\mathbf{G}_F = ([x, t_2], [i', t_3], [*, *])$ which is an allowable \mathbf{G}_F .

For the fault-free circuit, $L\{\tilde{\mathbf{G}}_F\}$ is the present output assigned to the gate. This is also true, with small modifications, for the case of the single stuck-at faults since any gate containing a stuck-at fault may be modeled simply by replacing the good gate with a faulty gate. The faulty gate will have some input or the output always set to logical 1 or 0. However, the situation is more complicated for shorted faults. Because of the action of the shorted faults, the free output of a gate which *drives* a shorted fault is not necessarily the constrained output of that gate. However, the action of the gate model is not affected since it produces the free output vector $\tilde{\mathbf{G}}_F$ which is independent of any shorted fault.

THE CIRCUIT MODEL

The model used to simulate the action of the interconnection of several gates is just as important as the gate model. Again the accuracy of the simulation depends on a good circuit model. The objective is to imitate the action of the actual circuit as closely as possible. This objective precludes the use of the Huffman Model or any schemes involving isolating feedbacks or leveling the circuitry. The model used here is both accurate and simple.

To simulate a fault-free asynchronous logic circuit, let the present time be t_0 and assume that the set of gates $\{G_i, i=1, 2, \dots, m\}$ are all the gates whose outputs are changing at time t_0 . Then the circuit can be modeled as follows:

1. Race analysis is performed for each flip-flop which is formed by two gates, both of which are in $\{G_i, i=1, 2, \dots, m\}$.
2. The new outputs are assigned to every gate in $\{G_i, i=1, 2, \dots, m\}$ by performing the assignment operation on the present free output vector of each G_i .
3. After all the new outputs of G_i have been assigned, the output of each gate $H_j, j=1, 2, \dots, n$, which is driven by any G_i whose output has changed, is calculated according to the gate model. If the output of some $H_k, 1 \leq k \leq n$, changed, then H_k is put into a list of gates whose output may change at time $t_0 + t_i$, where

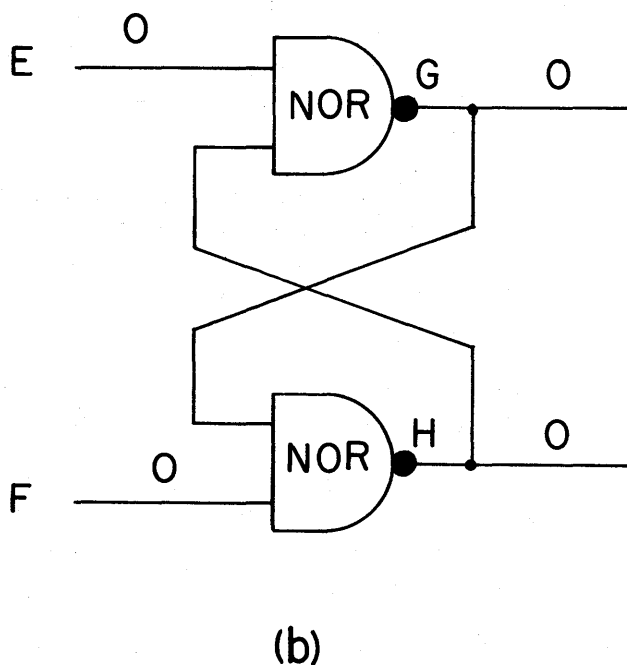
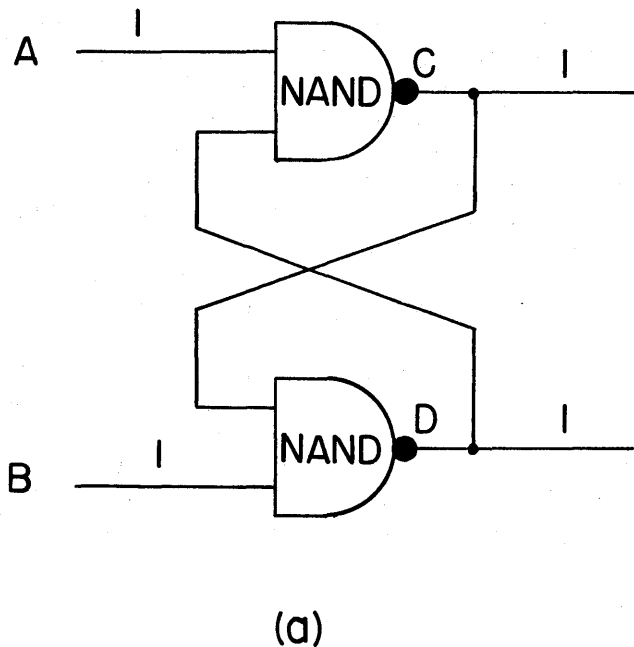


Figure 4(a)—Race conditions for a NAND flip-flop

Figure 4(b)—Race conditions for a NOR flip-flop

t_i is the transition time for H_k . If the output of H_k did not change, no further action is taken on the gate.

The important feature of this circuit model is that the gates H_j , $j=1, 2, \dots, n$, have their outputs calculated based on all the new values of the gates in $\{G_i, i=1, 2, \dots, m\}$. That is, every change which is going to occur at t_0 occurs before the output of any gate driven by any of the G_i is calculated. This process continues until there are no more gates whose outputs are scheduled to change after t_0 (and before any further primary input change). This model may cause the output of a gate to be calculated several times even in combinational circuitry. However, it can easily be seen that it provides an excellent simulation of the actual circuit operation.

The race analysis mentioned above is unrelated to the Huffman Model and is concerned with only the basic NAND and NOR flip-flops shown in Figure 4. If either flip-flop enters the state shown, then the next output of either flip-flop is unpredictable. An actual flip-flop will always settle and have one of its output terminals at logical 1 and the other at logical 0. However, which output terminal will be 0 is unpredictable. Therefore, if the race state is ever entered, both output terminals of the flip-flop are assigned the actual output value x to indicate the unpredictable output. This assignment is automatic if the flip-flop gates are ambiguous, but is performed by the race analysis if the gates are unambiguous.

If some arbitrary amount of simulated time has elapsed and the circuit still has not settled, then an oscillation is declared. When an oscillation is declared at time t_0 , every gate whose output is scheduled to change at some time greater than t_0 is assigned the output value x rather than its calculated output, and no other part of the simulation model is changed. The injection of the value x will eventually stabilize the circuit with some gates having the output x . This injection will be performed automatically if the oscillating gates are ambiguous, but will be performed by independent oscillation analysis if necessary.

The initial state of the circuit is an important feature of the circuit model. While the simulation may be started from some initial state other than the completely unknown state, where the output of every gate is the value x , this will probably not yield an accurate simulation. This is true because setting the state assumes that in the actual (perhaps faulty) circuit it is possible to set the state, and discards all information about the faults which would prevent the state from being set. Furthermore, even if the actual circuit has

been operating and should be in some known state, there is no guarantee that it will be in that state because of possible faults in the circuit. Therefore, the only way of guaranteeing an accurate simulation is to start with the output value of every gate set to the unknown value x and apply a homing sequence to set the state. The homing sequence will produce information about which faults will prevent the circuit from reaching the desired state. Then, in the actual circuit if the homing sequence fails to set the state of the circuit, one of the faults discovered above must exist in the circuit. This means that the homing sequence should be an integral part of the diagnostic tests for a logic circuit.

SIMULATION OF THE GOOD CIRCUIT

Sufficient information is now available to allow a simulation of the good machine to be performed. Each gate in the circuit may be assigned minimum and maximum transition delays. Based on the transition delays assigned to the gates, certain constrained hazards can be detected and a worst case timing analysis of the circuit can be produced.

The concept of the M -hazard¹ may be applied to a non-Huffman Model sequential circuit by simply observing that a constrained hazard is the possibility of a spurious pulse on the output of the circuit based on the specified gate time parameters. This general definition of a hazard is acceptable since the circuit output may be observed as the circuit settles allowing detection of static hazards (whether due to one input change or more is insignificant) and dynamic hazards. Since only detection of hazards is proposed, no distinction need be made between logic and function hazards. However, the constrained hazard is limited in that the possibility of the spurious pulse does not allow for any delay on any gate. The spurious pulse must be a real possibility in that the only delays which a gate may assume to allow the constrained hazard are those previously set by the timing parameters of a gate. For example, if a spurious pulse will occur if gate A has a delay of 10 nanoseconds, but the maximum transition delay allowed for gate A is 6 nanoseconds, then no spurious pulse will occur and no constrained hazard exists.

The constrained hazard detection can be achieved as follows. Consider a 2-input NAND gate G with inputs i and j , and let $k=1$ or 0 . If simulation produces a $k \rightarrow x \rightarrow k'$ (k to x to k') transition on the gate G , where $k \rightarrow x$ occurs at time t_1 and $x \rightarrow k'$ occurs at t_2 , then this means, because of the definition of x , that the output of the real gate G is changing from k to k' and the

change will occur between times t_1 and t_2 . The use of the x provides information on when $k \rightarrow k'$ may occur, and in fact provides worst case timing information based on the specified gate time parameters.

It can easily be seen from the NAND truth table shown in Figure 3 that if input $i=1$, then the gate G is sensitive to input changes on j , while if either input is logical 0, the output of G is always 1. Assume that in the actual circuit i makes a $k \rightarrow k'$ transition and $j=1$ when the gate G is in a steady state (the two inputs have not changed for at least the maximum transition delay time). Then, in the simulated circuit the present free output vector \bar{G}_F of G contains two * elements and (assuming there is ambiguity associated with the i transition) i is making a $k \rightarrow x \rightarrow k'$ transition, where $k \rightarrow x$ occurs at the earliest time t_1 when the actual i may change, and $x \rightarrow k'$ occurs at the latest time t_2 when the actual i may change. Assume that t_α and t_β represent the minimum and maximum $k \rightarrow k'$ transition delays of G , respectively. The effect of G will be to delay the $k \rightarrow k'$ transition and to either not change the time ($t_2 - t_1$) or to increase it because of ambiguity on G . That is, the output transition of G will be $k' \rightarrow x \rightarrow k$, where the $k' \rightarrow x$ occurs at $t_1 + t_\alpha$ and $x \rightarrow k$ occurs at $t_2 + t_\beta$. Since $t_\beta \geq t_\alpha$ by definition, the ambiguity region cannot decrease, and hence $(t_2 + t_\beta) - (t_1 + t_\alpha) \geq (t_2 - t_1)$. Therefore, the actual output transition of G will occur between $t_1 + t_\alpha$ and $t_2 + t_\beta$.

Also because of the definition of the x , any $k \rightarrow x \rightarrow k$ transition represents the possibility in the actual circuit of a $k \rightarrow k' \rightarrow k$ transition. Therefore, the $k \rightarrow x \rightarrow k$ transition represents a constrained hazard. The output of the simulated circuit may simply be observed, and any $k \rightarrow x \rightarrow k$ transition noted since this means, based on the specified timing parameters, the output of the actual circuit may experience a $k \rightarrow k' \rightarrow k$ transition.

Since each gate contains the worst case timing information, the simulation of the circuit produces the worst case timing analysis for the circuit based on the specified gate timing parameters. This means that if it is possible for a spurious pulse to occur in the actual circuit, then a $k \rightarrow x \rightarrow k$ pulse will occur in the simulated circuit. Since the simulation proceeds until the circuit has settled down, the effects of the $k \rightarrow x \rightarrow k$ pulse must be propagated forward. Then, if it is possible for the $k \rightarrow x \rightarrow k$ pulse to throw the sequential circuit into an unknown state, it will occur. Therefore, the effects of any hazard on the circuit being simulated are obvious since the state may become unknown. This worst case timing analysis can be extremely useful for investigating asynchronous circuits.

It is apparent that even a simulation of the good circuit using the models presented here can yield sig-

nificant information about the effects of gate delays which deviate from the normal. In general, a simulation of the good circuit should be performed before any fault simulation is attempted.

SIMULATION OF SHORTED FAULTS

If only the traditional single stuck-at-one and stuck-at-zero faults are considered, the effect of these faults on the model is minimal. A gate is simply treated as if it had one less input (for example, a NAND gate-input stuck-at-one) or as if its output were always logical 1 or 0 (for example, a NAND gate-input stuck-at-zero or gate-output stuck-at-one or -zero). For these faults only the gate model is affected, not the circuit model.

The simulation of single shorted faults (shorted input diodes or shorted gate-output nets) is more complex since both the gate model and circuit model are different in the presence of shorted faults. A new circuit model is required because of the bilateral effects of a shorted fault.

To allow discussion of the shorted faults, several definitions are helpful. A *node* is a point, where the possibility of a shorted input diode or shorted gate-output fault exists. No physical or electrical connection is necessary. For example, a node may be a gate or an integrated circuit crossover.

A gate *A* is said to *drive* a node *B* if the output of *A* is a part of node *B*. Similarly, a gate *A* is *driven* by a node *B* if some input of *A* is a part of node *B*. The effect of a shorted fault is to cause the output of some gate *G* to assume a value other than its free output.

Assume that the logic circuit under consideration has a more potent logical 0 than logical 1 so that if two gate-outputs are shorted together, any gate-output at 0 will force the shorted output to 0. Thus, if several terminals are shorted together, the resulting value on each terminal will be 1 if and only if all the terminals have the value 1 in the absence of the shorted fault (the free outputs are all 1). Similarly, the resulting value on each terminal will be 0 if one of the terminals would have the value 0 in the absence of the shorted fault. Therefore, the shorted fault performs the logic function AND on all the terminals driving the shorted node and forces each terminal to assume the resulting value. (A logic circuit with a more potent logical 1 will perform the logic function OR in the presence of a shorted fault.)

There are three cases of shorted faults which will be considered in detail. The most important case is the shorted gate-output fault. (Information about the physical layout of the circuit is necessary to obtain in-

formation about which gate-outputs might reasonably short together.) The other cases are the shorted diode faults on the logic gates performing the AND and OR functions. Define the *extended output net* of a gate *G* for a shorted fault *f* to be all the terminals in the circuit which can force the output of *G* to change from 1 to 0. Three cases of the extended output net will be considered here. The first case is for the shorted gate-outputs, the second is for the shorted input diode in an AND gate and the third is for the shorted input diode in an OR gate.

Consider the gates $G_i, i = 1, 2, \dots, m$, such that all G_i drive the shorted fault node *s* as shown in Figure 5. Each G_i may drive several shorted fault nodes. If the fault corresponding to node *s* exists, then the output terminals of all the G_i will be connected together. Therefore, the extended output net of some gate $G_k, 1 \leq k \leq m$, for the shorted fault *s* contains all the output terminals of the gates $G_i, i = 1, 2, \dots, m$. Then, the actual output g_k of G_k is

$$g_k = \prod_{i=1}^m g_i \text{ for } k = 1, 2, \dots, m, \quad (1)$$

where the \prod means the logical AND operation. If the output of some gate $G_k, 1 \leq k \leq m$, changes from 1 to 0 because of the shorted fault, then every gate which is driven by G_k must be reevaluated except the shorted node *s*. It is not necessary to reevaluate node *s* since

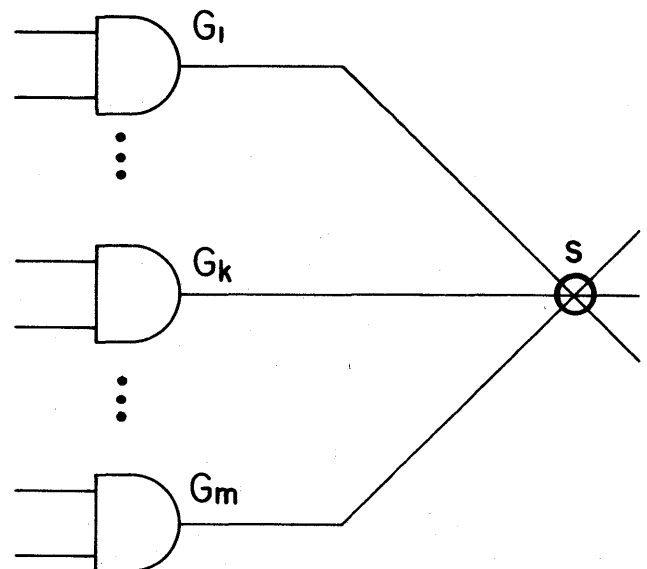


Figure 5—Shorted gate output terminals

the logic value on s will not change as a result of the change of g_k . This can easily be seen as follows. In order for the output g_k of G_k to change, s must have changed to logical 0 or x because (1) produced a 0 or x output. Since (1) can produce a 0 output only if some $g_h = 0$ for $1 \leq h \leq m$ already, adding another 0 input g_k cannot affect (1). Similarly, (1) can produce an x output only if there are no 0 inputs and at least one x input. Then adding another x input will not affect (1).

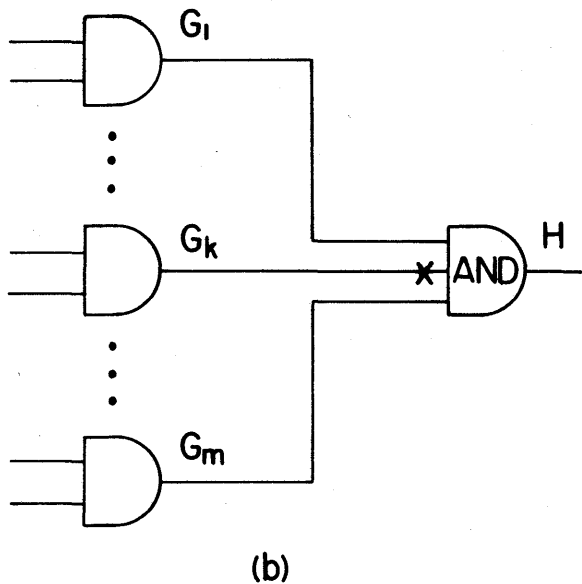
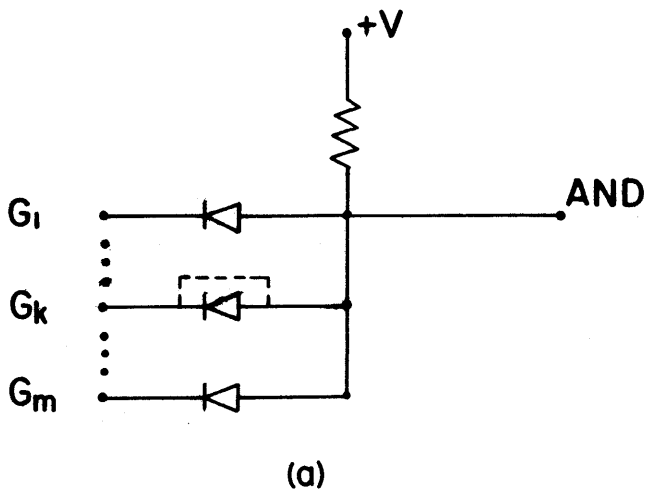


Figure 6(a)—Diodes performing AND function

Figure 6(b)—Shorted diode for an AND gate

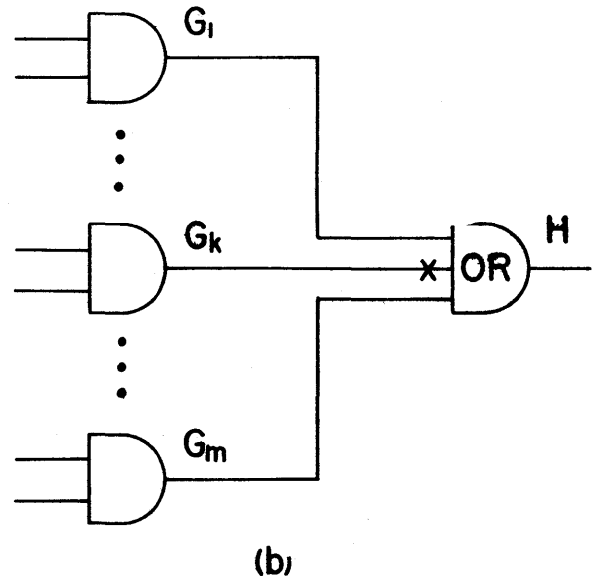
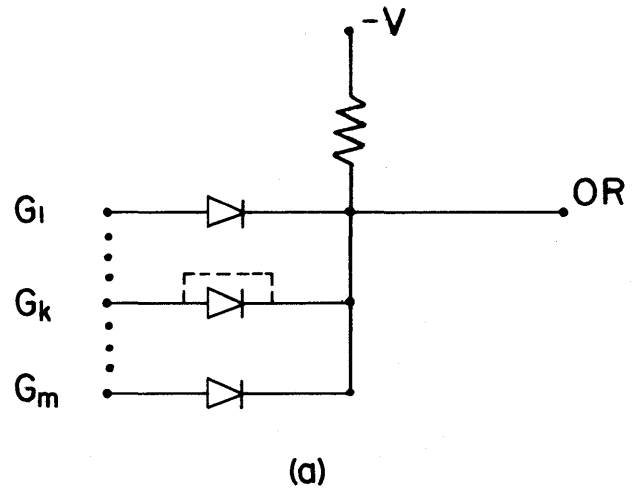


Figure 7(a)—Diodes performing OR function

Figure 7(b)—Shorted diode for an OR gate

The problem of a shorted input diode in the AND (or NAND) gate as shown in Figure 6(a) is similar. For the circuit shown in Figure 6(b), assume the AND gate H has a shorted input diode which is driven by the gate G_k , $1 \leq k \leq m$. If any input g_h , $1 \leq h \leq m$, is a logical 0, then g_h will force the output g_k of G_k to 0, but will not affect the output of any other G_h for $h \neq k$. The extended output net of G_k for the shorted input diode of H includes the output terminal of every G_i , $i = 1, 2, \dots, m$. Therefore, the output of gate G_k is

given by

$$g_k = \prod_{i=1}^m g_i \quad (2)$$

Notice that the output of gate H is unaffected by the shorted diode. The only gates which may be affected are the gates other than H driven by G_k .

For the previous two cases, the gates which drive the shorted fault have been affected by the fault. However, this is not the case for the shorted diode in the OR gate as shown in Figure 7(a) and 7(b). In this case if $g_k=0$, $1 \leq k \leq m$, then the gate H assumes the output value 0 and all g_i , $i=1, 2, \dots, m$, are forced to 0. Therefore, the extended output net of each G_i for the shorted input diode on H from G_k contains the output terminal of G_i and the output terminal of G_k . Consequently the output g_i of the G_i is given by

$$g_i = g_i \cdot g_k \quad \text{for all } i=1, 2, \dots, m \quad (3)$$

Here the output of H may change if some $g_k=1$ for $1 \leq h \leq m$ and $h \neq k$. In addition, the inputs with logical 1 will be forced to 0 when $g_k=0$. Therefore, any gate driven by H or G_i , $1 \leq i \leq m$, is affected if H or G_i changes its value.

Next, the circuit model necessary to simulate the shorted faults must be presented. It is assumed that if the actual output of a gate G is forced to some value other than the fault-free output of G , the transition occurs instantaneously and does not affect the free output vector of the gate model G . Similarly, it is assumed that if the effect of the fault is removed, the actual gate G will recover instantaneously to its fault-free output value.

Assume that some set of gates $\{F_i, i=1, 2, \dots, m\}$ drive a shorted fault node s . Furthermore, assume that some set of gates $\{G_j, j=1, 2, \dots, n\}$ have outputs which are to change at the present time t_0 . Suppose that the output of gate G_j , $1 \leq j \leq n$, is identical to the output of the gate F_i , $1 \leq i \leq m$. Then the model used for simulating shorted faults is as follows:

1. For every gate G_j assign the new output to G_j without regard to any shorted faults.
2. Perform race analysis for each flip-flop which is formed by two gates, both of which are in $\{G_j, j=1, 2, \dots, n\}$.
3. If any gate G_j , $1 \leq j \leq n$, drives the shorted fault node s , form the new output for all the F_i 's as described before based on the free outputs of the F_i 's. Then assign the new outputs to the F_i 's.
4. For each gate G_j , $1 \leq j \leq n$, and F_i , $1 \leq i \leq m$, if the output of G_j or F_i changed, then calculate the new output of each gate driven by G_j or F_i .

The model presented yields the correct timing for the circuit since the effect of a shorted fault is generated and propagated as soon as it occurs. That is, the reverse effect is immediately accounted for as it should be. Also, since the free and constrained outputs are isolated, the insertion of the shorted faults still allows the worst case timing analysis, even of the faulty circuit.

It is possible for the output of some gate G to be k , where $k=1$ or 0 for the good circuit, but to be x under some fault condition f . This means that G may or may not detect the presence of fault f . It is also possible for the output of G to be the value x for some fault f but to have $G=k$ for the good circuit.

DISCUSSION

The simulator proposed here provides an extremely accurate simulation of any large asynchronous sequential logic circuit. Both the good circuit and the faulty circuit containing shorted gate-outputs and shorted input diodes can be simulated. Although only the simulation of shorted faults has been discussed, it is obvious that the traditional stuck-at-one and stuck-at-zero faults can easily be simulated. The gate model used allows the detection of constrained hazards as well as the worst case timing analysis of the circuit.

This simulator has been implemented, on the IBM/360 TSS system which is a virtual memory, conversational access system operating on Model 67 computers. The simulator is written in 360 Assembler Language and uses approximately 25K bytes of core for program storage. Additional storage is necessary for the various tables used.

During evaluation tests, circuits of up to 48,000 gates have been simulated. Speeds of up to 100 microseconds/vector·fault·gate have been achieved for highly sequential circuits, where each gate is ambiguous. It is difficult to compare the simulation time to that of any existing simulator because no existing systems have the same power as the one presented here. In order to make some comparison, let us consider the latest simulator⁵ which required about the same simulation time, but used nonambiguous gates, no high-frequency rejection and no flip-flop race analysis. However, the simulation time of this latest simulator will be much longer even if only the consideration of ambiguous gates is added.

REFERENCES

- 1 E B EICHELBERGER
Hazard detection in combinational and sequential circuits
IBM Journal of Research and Development Vol 9 No 2
pp 90-99 1965

-
- 2 M YOELI S RINON
Application of ternary algebra to the study of static hazards
Journal of ACM Vol 11 No 1 pp 84-97 1964
- 3 S SESHU
The logic organizer and diagnosis programs
Report R-226 Coordinated Science Laboratory University
of Illinois Urbana Illinois 1964
- 4 H Y CHANG
*A method for digitally simulating shorted input diode
failures*
Bell Systems Technical Journal Vol 48 No 4
pp 1957-1966 1969
- 5 S A SZYGENDA D W ROUSE E W THOMPSON
*A model and implementation of a universal time delay
simulator for large digital nets*
Proceedings of AFIPS Spring Joint Computer Conference
pp 207-216 1970
- 6 R J DIEPHUSIS
Logic design assisted by interactive computer simulation
Digest of Northeast Electronics Research and Engineering
Meeting 1970
- 7 I H YETTER
*High speed fault simulation for Univac 1107 computer
system*
Proceedings of ACM National Conference pp 265-277
1968
- 8 G G HAYS
Computer-aided design: simulation of digital design logic
IEEE Transactions on Computers Vol C-18 No 1 pp 1-10
1969
- 9 J S JEPHSON R P MC QUARRIE
R E VOGELSBERG
A three value computer design verification system
IBM System Journal Vol 8 No 3 pp 178-188 1969

Adaptive memory trackers

by G. EPSTEIN

ITT Gilfillan Incorporated
Van Nuys, California

INTRODUCTION

This introduces a mathematical model for a control process called an adaptive memory tracker. The control or adaptive variable is the memory length (i.e., retention time) of a filter model, rather than other parameters within the filter model. The adaptation in memory length is achieved through the use of two different types of filters which are based on this filter model. These filters are called shrinking memory and penetrating memory recursive filters, and their application as a function of an error signal decreases or increases, respectively, the memory length of the filter model. An example of this is shown diagrammatically in Figure 1.

Such an approach is of interest from the standpoint of implementation in a mechanico-electrical system because it allows the tracker to base its performance only on relevant input data—that is, the memory length of the filter model adapts so as to ignore input data before an adjusted point in time. This contrasts with other trackers^{1,2} in which all past input data are incorporated, through adjustment of exponential damping on this data as a function of adaptive parameters within the filter model.

The mathematical model is of interest from a cybernetics standpoint because it isolates the operation of memory and allows the effect of adaptive changes in memory length to be an object of study.

Elementary examples and simulations are given. These display not only properties of the model, but show, in a functional sense, that the model pertains to other kinds of systems, such as biological or societal systems.^{3,4} The model pertains to these other systems to the extent that the responses of the trackers for the given examples and simulations are similar to the responses of these other systems. It is indicated that more sophisticated examples may be required to realize the full value of the model.

It is required that the above systems be conceived as sensing input data and having memory.⁵ It is further

required that the input data be arranged in a time sequence, and that these input data be stored in memory in a way which preserves this time sequence.

An example of a mechanico-electrical system is a radar and computer complex, or group of such complexes. An example of a biological system is an organism, and an example of a societal system is a group of such organisms.

The inputs are conceived as units, denoted by u_n , $n=1, 2, \dots$. For simplicity, it will be assumed that the time interval, T , between these units is constant. The units are obtained from a physical model, whose values are denoted by p_n , subject to disturbances, whose values are denoted by d_n . Thus, $u_n = p_n + d_n$, $n=1, 2, \dots$, as shown in Figure 1.

Each unit u_n is stored in memory in the form of $m(n)$ components. In the mechanico-electrical system, there are $m(n)$ addresses in computer memory, each address consisting of a certain number of bits (the word length), and each u_n is a linear combination of its $m(n)$ components (obtained from $m(n)$ measurements) stored in these addresses. In the biological system, the components involve the synapses in a neuronal pattern, and possibly elements of hormonal, glial, or other nature. In the societal system, the latter $m(n)$ components are extended over a number of organisms.

Thus, the memory of the system may be of an aggregate nature, in the sense that the system may consist of a group of radar and computer complexes, or a group of organisms, each with its own memory. It is required, however, as mentioned above, that the time sequence of the input data conceived as units, u_n , be preserved through the storage of their $m(n)$ components in memory. In the biological and societal systems, therefore, memory is construed as being short or medium term⁶ and temporal,⁷ for these types of memory seem to be time dependent in character, whereas long term memory⁶ and categorical memory⁷ seem to be more of an attention focusing character. Furthermore, in these latter systems, memory is construed as being

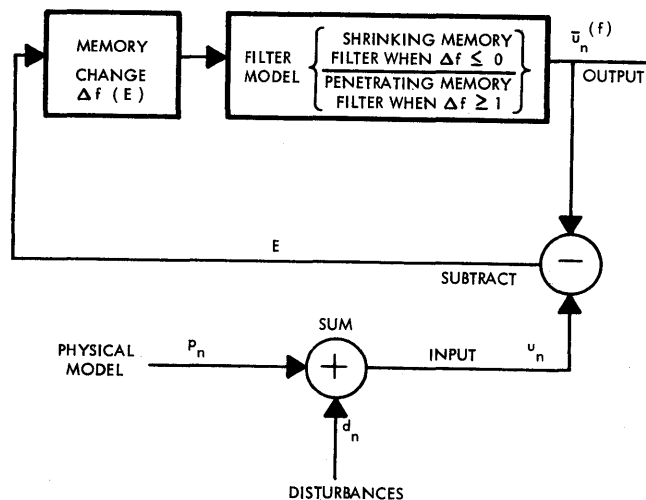


Figure 1—Adaptive memory tracker for case $E = (\bar{u}_n^{(f)} - u_n)$

internal, within the organism(s), rather than external, in the form of artifacts, such as records, structures, etc.

The filter model is of order $N+1$ with outputs $\bar{u}_{dn}^{(f)}$, $d=0, 1, 2, \dots, N$. The superscript (f) is the adaptive variable, the memory length being $(f-1)T$; i.e., the outputs may be written in the form:

$$\bar{u}_{dn}^{(f)} = \sum_{i=n-f+1}^n W_{d(i-n+f)} u_i, \quad d=0, 1, \dots, N; f \geq 2. \quad (1)$$

The subscript d denotes the $N+1$ outputs and the subscript n is the recursive parameter. The case $d=0$ is designated as the position output, $\bar{u}_{0n}^{(f)} = \bar{u}_n^{(f)}$.

The error signal in position at the n th step is:

$$E = \sum_{i=1}^e \epsilon_i (\bar{u}_{n-i+1}^{(f(n-i+1))} - u_{n-i+1}). \quad (2)$$

The change in the adaptive variable f at the n th step is denoted by Δf , and is given as an integral valued function of the error signal, $\Delta f(E)$.

The tracker employs a shrinking memory recursive filter when $\Delta f = -s \leq 0$, and a penetrating memory recursive filter when $\Delta f = p+1 \geq 1$.

If the output at the $(n-1)$ st step is $\bar{u}_{i(n-1)}^{(v)}$, $i=0, 1, \dots, N$, then the shrinking memory filter is:

$$\bar{u}_{dn}^{(v-s)} = \sum_{i=0}^N M_{di} \bar{u}_{i(n-1)}^{(v)} + N_d u_n + \sum_{j=0}^s S_{dj} u_{n-f+j}, \quad d=0, 1, \dots, N, \quad (3)$$

and the penetrating memory filter is:

$$\bar{u}_{dn}^{(v+p+1)} = \sum_{i=0}^N J_{di} \bar{u}_{i(n-1)}^{(v)} + K_d u_n + \sum_{j=1}^p P_{dj} u_{n-f-j}, \quad d=0, 1, \dots, N. \quad (4)$$

The matrix coefficients M_{di} , N_d , S_{dj} , depend on the filter model, v and s . The matrix coefficients J_{di} , K_d , P_{dj} , depend on the filter model, v and p . These matrix coefficients are chosen, of course, so that (1) is satisfied.

It should be noted that when $s=0$, the shrinking memory filter becomes the well known finite memory filter⁸ and when $p=0$, the penetrating memory filter becomes the well known growing memory filter,⁹ of which the Kalman filter¹⁰ is an example.

If there is an F such that $f \leq F$ for all f , the adaptive memory tracker is bounded; i.e., the memory length does not exceed $(F-1)T$. Otherwise, the adaptive memory tracker is unbounded.

To illustrate in a simple way, let $e = \epsilon_1 = 1$, so that the error signal is just:

$$E = \bar{u}_n^{(f)} - u_n. \quad (5)$$

This case is shown in Figure 1.

To further simplify, let the filter model be a straight line, least squares model; that is, $N=1$, and $\bar{u}_n^{(f)}$ is the endpoint of a straight line such that the square of the differences between this line and the last f inputs is a minimum. The disturbances d_n are chosen to be independent samples of normally distributed noise of mean 0 and constant standard of deviation σ .

The effect of adaptive changes in memory length is perhaps best shown by choosing $\Delta f(E)$ to be two valued. This is also the easiest to implement in a computer. Specifically,

$$\Delta f = +1 (p=0) \quad \text{when} \quad |E| \leq K\sigma \quad (6)$$

and

$$\Delta f = -1 (s=1) \quad \text{when} \quad |E| > K\sigma, \quad (7)$$

where K is a constant. This is shown in Figure 2.

The penetrating memory filter for this case is the well-known growing memory least squares filter:¹¹

$$\begin{aligned} \bar{u}_n^{(v+1)} &= \bar{u}_{0n}^{(v+1)} \\ &= \frac{v(v-1)}{(v+1)(v+2)} \bar{u}_{0(n-1)}^{(v)} \\ &\quad + \frac{v(v-1)}{(v+1)(v+2)} u_{1(n-1)}^{(v)} + \frac{2(2v+1)}{(v+1)(v+2)} u_n \\ \bar{u}_n^{(v+1)} \cdot T &= \bar{u}_{1n}^{(v+1)} \\ &= -\frac{6}{(v+1)(v+2)} \bar{u}_{0(n-1)}^{(v)} \\ &\quad + \frac{(v-1)(v+4)}{(v+1)(v+2)} \bar{u}_{1(n-1)}^{(v)} + \frac{6}{(v+1)(v+2)} u_n \end{aligned} \quad (8)$$

The shrinking memory filter for this case may be

derived from (8) and the equations for the finite memory least squares filter.¹¹ The result is:

$$\begin{aligned}
 \bar{u}_n^{(v-1)} &= \bar{u}_{0n}^{(v-1)} \\
 &= \frac{(v-9)}{(v-1)} \bar{u}_{0(n-1)}^{(v)} + 5\bar{u}_{1(n-1)}^{(v)} + \frac{2(2v-3)}{v(v-1)} u_n \\
 &\quad + \frac{2}{(v-1)} u_{n-v+1} + \frac{2(v+3)}{v(v-1)} u_{n-v} \\
 \bar{u}_n^{(v-1)} \cdot T &= \bar{u}_{1n}^{(v-1)} \\
 &= -\frac{18}{(v-1)(v-2)} \bar{u}_{0(n-1)}^{(v)} + \frac{(v+10)}{(v-2)} \bar{u}_{1(n-1)}^{(v)} \\
 &\quad + \frac{6}{v(v-1)} u_n + \frac{6}{(v-1)(v-2)} u_{n-v+1} \\
 &\quad + \frac{6(v+2)}{v(v-1)(v-2)} u_{n-v}. \tag{9}
 \end{aligned}$$

In these equations, $\bar{u}_n^{(f)} = \bar{u}_{1n}^{(f)}/T$ denotes the velocity output.

The figures which follow show the results of simulations for this adaptive memory tracker on an IBM 1130 computer, using disturbances d_n which are independent samples from a pseudo normal distribution generated by a subroutine within this computer. In these figures, the values p_n which constitute the physical model are shown by a dashed line, the inputs $u_n = p_n + d_n$ are shown by crosses, and the tracker outputs $\bar{u}_n^{(f)}$ are shown by the solid line. At the top of each figure is a plot of the values that the adaptive variable f assumes for each value of n .

A convenient measure of the tracker response as a function of f when the physical model is strictly linear is the noise-reduction-ratio.¹¹ This may be used, therefore, in Figure 3, which follows, but only rarely in the remaining figures.

A simple measurement of the tracker response over

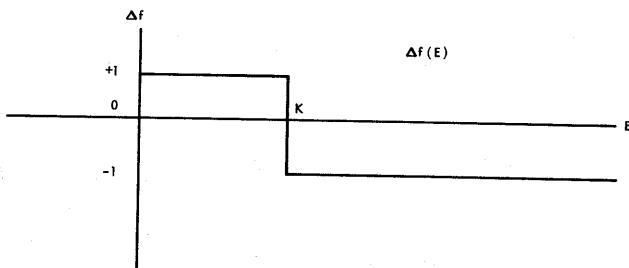


Figure 2—Change in memory length (Δf) as a function of error signal (E) for trackers shown in Figures 3, 4, and 5

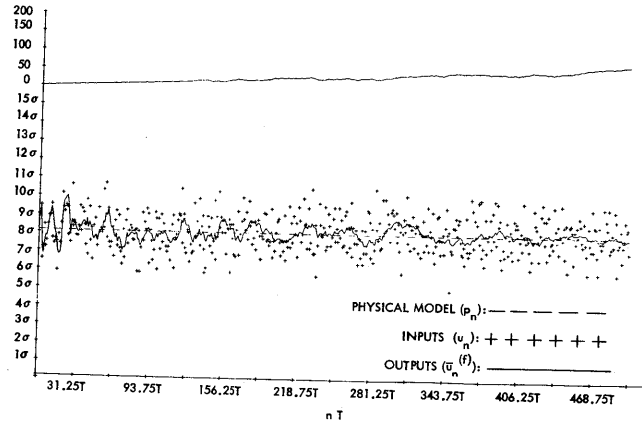


Figure 3—Adaptive memory, straight line, least squares tracker when physical model is constant, $\Delta f(E)$ given by Figure 2, with $K=0.8$

the interval $i=j$ to $i=n$ is given by the response:

$$R_{j,n} = \frac{\sum_{i=j}^n |\bar{u}_i^{(f(i))} - p_i|}{n-j+1}. \tag{10}$$

The response is viewed visually in an easy way in these figures as $R_{n,n} = |\bar{u}_n^{(f)} - p_n|$, the magnitude of the difference between the solid line representing the tracker output $\bar{u}_n^{(f)}$, and the dashed line representing the physical model p_n at each value of n . Values of the response $R_{j,n}$ over particular intervals will lend preciseness to the description of the tracker response.

Figure 3 shows the case where the physical model is constant and $K=0.8$. This may be called a regulator, homeostat,¹² or, even, sociostat,¹³ where the control variable is memory length. The slow increase in f and resulting slow improvement in the tracker response are due to the low value of $K=0.8$.

In Figures 4 and 5, the value of K is $K=1.0$, and the physical model is piecewise linear, over the path A, B, C, D, E, F, G . It is assumed that BA is extended before A linearly, and the tracker is started at A with the value of f as shown. The operation of the tracker before A is similar to that shown in Figure 3. This physical model corresponds to the case where there are repeated changes in environmental constraints. It may be produced in an approximate sense by an aircraft making repeated changes in bearing of 90° through high g maneuvers, or through repeated changes in laws, either in a society or, as in learning reversal experiments, on organisms.¹⁴

In Figure 5, the tracker does not track $BC, CD, DE, EF,$ and FG , until $B_1, C_1, D_1, E_1,$ and F_1 , respectively,

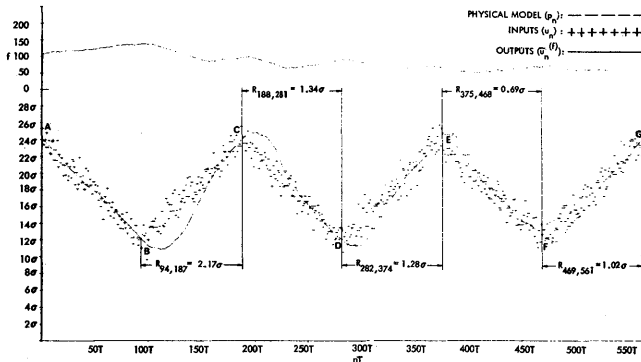


Figure 4—Adaptive memory, straight line, least squares tracker when physical model is piecewise linear, $\Delta f(E)$ given by Figure 2, with $K = 1.0$

for one of two reasons:

Either

- (i) The tracker does not sense data from the physical model BB_1, CC_1, DD_1, EE_1 , and FF_1 , and in the absence of data simulates data from the substitute model BB_0, CC_0, DD_0, EE_0 , and FF_0 ; or
- (ii) The tracker is being deceived and receives false data from the substitute model BB_0, CC_0, DD_0, EE_0 , and FF_0 .

In either case, the tracker senses data from the physical model starting at B_1, C_1, D_1, E_1 , and F_1 . The values s_n of the substitute model are shown in Figure 5 as a dotted line, so that the values of the input data u_n along the paths $BB_0, CC_0, DD_0, EE_0, FF_0$, are given by $u_n = s_n + d_n$.

It can be seen that this tracker shows very good learning characteristics. Not only does the tracker show

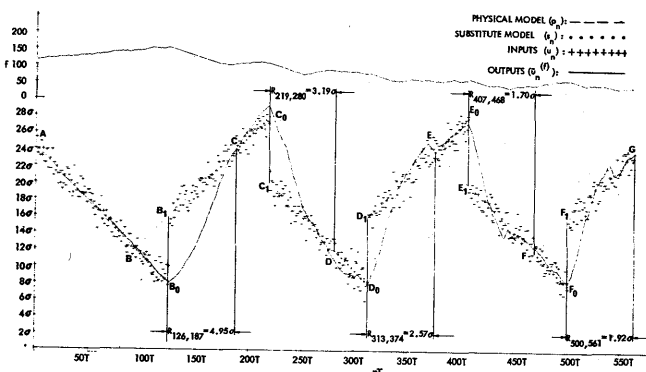


Figure 5—Adaptive memory, straight line, least squares tracker when physical model is piecewise linear with substitute, $\Delta f(E)$ given by Figure 2, with $K = 1.0$

improved response to each change in path of the physical model, but the tracker also distinguishes between the conditions of Figures 4 and 5, in the former case stabilizing to values of f in the range $52 \leq f \leq 76$, and in the latter case stabilizing to values of f in the range of $40 \leq f \leq 65$. It can be seen that in Figure 4 the response improves from $R_{94,187} = 2.17\sigma$ to $R_{375,468} = 0.69\sigma$ and $R_{469,561} = 1.02\sigma$. In Figure 5 the response improves from $R_{126,187} = 4.95\sigma$ to $R_{407,468} = 1.70\sigma$ and $R_{500,561} = 1.92\sigma$. The improvement in response shown by this simple tracker is similar to that shown by the higher vertebrates.¹⁴ Such performance, obtained with the filter model, $K = 1, e = \epsilon_1 = 1, s = 1, p = 0$, all being fixed, suggests that more sophisticated examples, where some of these are allowed to vary, either in a preset way or adaptively, may be of more value for future work and research.

In the above examples, the adaptive memory tracker is unbounded. Practical implementation within a computer (whose physical memory is finite) requires a value, F , so that $f \leq F$ for all f .^{*} This is easily accomplished with the above tracker by adding the following proviso to (6) and (7).

$$\text{If } v = F \text{ then } \Delta f = -1 (s = 1) \text{ regardless of } |E|; \text{ i.e., in this case, the value of } |E| \text{ is ignored and (9) is used.} \quad (11)$$

To give a more sophisticated example, consider the same filter model (straight line, least squares), the same $e = \epsilon_1 = 1$, but $\Delta f(E)$, the change in memory length as a function of error signal, given as shown in Figure 6. Here, $\Delta f(E)$ is again a fixed function, but now Δf lies in the range $-50 \leq \Delta f \leq +50$, as shown. This implies

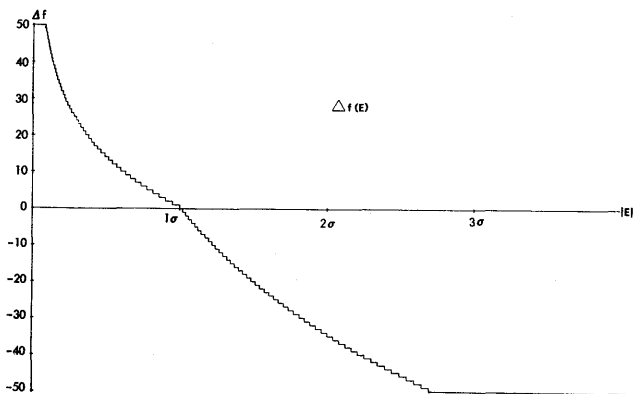


Figure 6—Change in memory length (Δf) as a function of error signal (E) for tracker shown in Figures 7 and 8

^{*}Of course there is no need from the standpoint of performance to exceed the required specifications for the problem at hand.

values of $s=0, 1, \dots, 50$ and values of $p=0, 1, \dots, 49$. This is of little interest from an implementation standpoint, and the corresponding equations for the shrinking memory and penetrating memory filters need not be given. This may be of interest, however, for biological systems. It is reasonable to suppose that there is more flexibility in such systems than implied by the binary alternative shown in Figure 2. One purpose of this example is to show that large variations in memory length as a function of error signal have unfortunate consequences in terms of tracker response. In particular, such variability in short or medium term temporal memory occurs in some cases of hysteria, or amnesia in senile dementia. At certain times of day, a patient's temporal memory extends over a period of hours; at other times of day, the patient's temporal memory extends only over a period of minutes.

Figures 7 and 8 demonstrate the operation of this tracker for the same cases as presented in Figures 4 and 5. Here the tracker is bounded with $F=175$ (corresponding to a form of permanent amnesia). The values of $\Delta f(E)$ given by Figure 6 are limited by the proviso that f lie in the range $2 \leq f \leq 175$.

As in these earlier figures, the line BA is extended linearly before A , in order to provide previous input data. In Figures 7 and 8, however, the tracker is started normally at $f=2$.

The tracker response over the path AB indicates that such a tracker is a satisfactory homeostat as long as F is sufficiently high. There are sudden sharp drops in f , which would seriously degrade the tracker response were the value of F too low. The chosen value of $F=175$ is sufficiently high to prevent this degradation.

However, the response of the tracker to repeated changes in environmental constraints is poor. In Figure 7 the response values over the indicated intervals vary

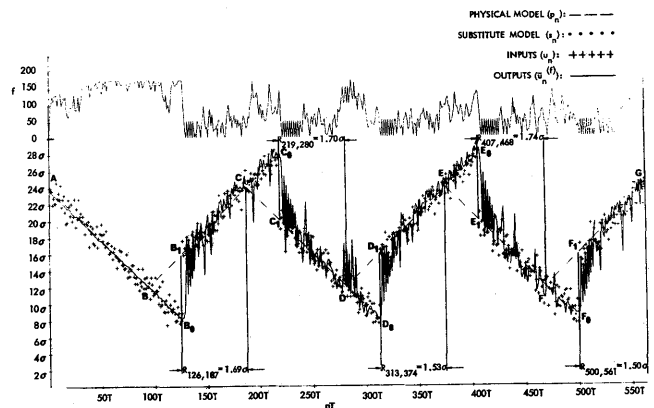


Figure 8—Adaptive memory, straight line, least squares tracker when physical model is piecewise linear with substitute $\Delta f(E)$ given by Figure 6, with $F=175$

between 0.70σ and 0.98σ ; in Figure 8 the response values over the indicated intervals vary between 1.50σ and 1.74σ . There is no improvement in the response, such as shown by the decrease in response values of Figures 4 or 5. The response values of Figures 7 and 8 are of the same order as the final response values of Figures 4 and 5, respectively, but here the response is very erratic, with many sharp increases in $R_{n,n}$ over all values of n after B in Figure 7 and B_1 in Figure 8.

Thus the response of the tracker to repeated changes in environmental constraints is poor in Figure 7, and worse in Figure 8. This erratic response with no learning is similar to the response of the above mentioned patients in such a situation as, for example, repeated transfers from one hospital to another. The response of these patients in this case may, in fact, become so poor as to result in death.

In all of the above, including the simulations, the model or plant error¹⁵ is zero. It follows from^{8,16} that the implementation of an adaptive memory tracker which uses a least squares recursive filter model would result in non-zero model or plant errors accumulating without bound as the number of recursions increases. This may be avoided either by considering other filter models, such as the stable filter models indicated in References 8 and 17, for example, or by restarting the adaptive memory tracker at regular intervals before these errors become too great. Note in particular that this tracker does not distinguish between the accumulation of errors in \bar{u}_n^f and the variation in u_n caused by the disturbances d_j , $j=n, n-1, \dots, n-e+1$, as shown by the equation for the error signal (2). Thus, as n increases, an accumulation of such errors in \bar{u}_n^f which exceeds the variations in u_n caused by the disturbances will result in increases in E . Since adaptive memory

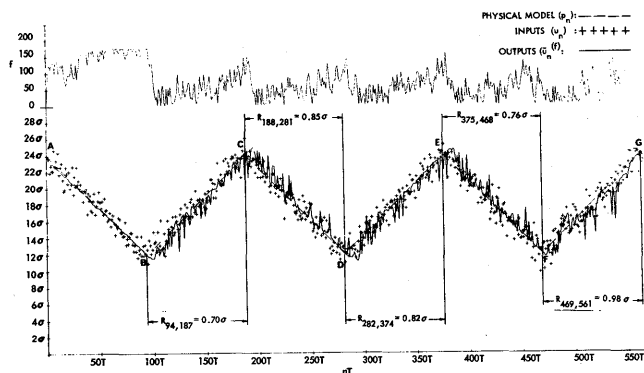


Figure 7—Adaptive memory, straight line, least squares tracker when physical model is piecewise linear, $\Delta f(E)$ given by Figure 6, with $F=175$

trackers in normal operation have $\Delta f(E)$ as a monotonically decreasing function*, it follows that the value of the adaptive variable f will eventually reduce to $f=2$, its minimum value, at which point the tracker can be restarted.

The restart equations for the above examples at $n = n_c$ are just

$$\begin{aligned}\bar{u}_{n_c}^{(2)} &= \bar{u}_{0n_c}^{(2)} = u_{n_c} \\ \bar{u}_{n_c}^{(2)} \cdot T &= \bar{u}_{1n_c}^{(2)} = u_{n_c} - u_{n_c-1}\end{aligned}\quad (12)$$

In the above discussion, only simple examples have been considered, in order to display basic values of the mathematical model in an easy way and, in particular, to highlight the effect of adaptive changes in memory length, all other parameters being fixed. It seems clear that the mathematical model will have more value when other parameters, such as $\Delta f(E)$, or e and e_i , are allowed to vary, and especially when the physical models or the statistics of the disturbances are more complicated. It is recommended that these ideas be extended and applied.

ACKNOWLEDGMENT

Valuable assistance was provided by Lowell Dean McMahan, who performed the programming for the simulations.

REFERENCES

- 1 T R BENEDICT G W BORDNER
Synthesis of an optimal set of radar track-while-scan smoothing equations
IRE Transactions on Automatic Control pp 27-36
July 1962
- 2 H R SIMPSON
Performance measures and optimization condition for a third-order sampled-data tracker
IEEE Transactions on Automatic Control pp 182-183
April 1963

- 3 E NAGEL
The structure of science—Problems in the logic of scientific explanation
Harcourt Brace and World 1961
- 4 L APOSTEL
Towards the formal study of models in the non-formal sciences in the concept and the role of the model in mathematics and natural social sciences
Editor H Freudenthal Proceedings of the Colloquium Sponsored by the Division of Philosophy of Sciences organized at Utrecht January 1960 by H. Freudenthal
Reidel Dordrecht 1961
- 5 W R ASHBY
An introduction to cybernetics
Chapman and Hall Ltd 1956
- 6 D A NORMAN Editor
Models of human memory
Academic Press 1970
- 7 J M NIELSON
Memory and amnesia
San Lucas Press 1958
- 8 G EPSTEIN
On finite—Memory recursive filters
IEEE Transactions on Information Theory Vol IT 16
No 4 pp 486-487 July 1970
- 9 M BLUM
Recursion formulas for growing memory digital filters
IRE Transactions on Information Theory Vol IT 4
pp 24-30 March 1958
- 10 R E KALMAN
A new approach to linear filtering and prediction problems
Journal of Basic Engineering Vol 82D pp 35-45 March 1960
- 11 N LEVINE
A new technique for increasing the flexibility of recursive least squares data smoothing
The Bell System Technical Journal pp 821-840 May 1961
- 12 W B CANNON
The wisdom of the body
NY 1932
- 13 A L STINCHCOMBE
Constricting social theories
Harcourt Brace and World 1968
- 14 M E BITTERMAN
The evolution of intelligence
Scientific American Vol 212 No 1 pp 92-100 January 1965
- 15 H W SORENSON
Least-squares estimation from Gauss to Kalman
IEEE Spectrum pp 63-68 July 1970
- 16 G EPSTEIN
Comment on 'on finite-memory recursive filters'
IEEE Transactions on Information Theory Vol IT 17
No 5 September 1971
- 17 G EPSTEIN
A note on the derivation of finite-memory almost-least-squares recursive filters
IEEE Transactions on Information Theory Vol IT-17
No 6 November 1971

* It may be of interest to consider functions, $\Delta f(E)$, which are not monotonically decreasing, in order to study organisms whose memory operation is of an abnormal or so-called paradoxical type.

A panel session—Planingg community information utilities

Conference Results

by BARRY W. BOEHM

The RAND Corporation
Santa Monica, California

Although diversity of opinion characterized many of the individual discussions, the conference yielded a surprisingly strong degree of consensus on a series of four major related points.

1. *Mass information utilities of some sort will be with us by the 1980s.* They will probably be based on cable-TV to the home and a low data-rate return line, although some participants felt that two-way video, and particularly Picturephone, offered a strong alternative. Some precursors exist now in airline and ticket reservation systems, IBM's Advanced Administrative System, and Mitre's TICCET system for elementary education in Reston, Va. Some commercial planning is going on, including efforts at Hughes and RCA, and a multi-client study by A. D. Little. Paul Baran cited a market analysis by the Institute for the Future, indicating very little large-scale penetration before the 1980s and about a \$15-20 billion market by the end of the 1980s.

2. *Mass information utilities carry a great deal of social risk.* Especially within the tight constraints of maintaining economic self-sufficiency, it will be difficult to avoid effectively discriminatory service policies between rich and poor users, urban and rural users, English-speaking and non-English-speaking users. Thus, any explicit or implicit public support of such utilities will have to be carefully thought through. An information utility will probably widen the gap between the information-rich and the information-poor, although probably not in the long run. Telepurchasing and continuous credit would increase the temptations and hazards of overspending, but could on the other hand eliminate the overcharging for goods in ghetto stores.

Information privacy aspects will certainly be touchy, even though economics will probably dictate a decentralized file structure. The polling and voting area is particularly sensitive to the quality of safeguards. Pro-

vision for an "Information Bill of Rights" is certainly necessary, and perhaps also for such things as anonymous coin-operated terminals and a "Fifth Amendment" switch (to insure user anonymity) on the console. Management aspects of the utility will be just as touchy. Even if the utility were just a distributor, the management would have considerable power over priorities. And, if users strongly adjust their lives around the utility, they won't have much choice but to go along with the management.

3. *A well-designed, scientifically-evaluated Prototype Community Information Utility (PCIU) would greatly reduce the long-term social risk.* It would provide an opportunity to sense the resulting social strains and experiment with ways to reduce or eliminate them. It would also reduce the economic risk to business in interfacing with an information utility. As Bruce Gilchrist expressed it, a PCIU could provide the kind of future socioeconomic insights we might have gained by equipping a representative community in 1920 with two cars per family and the associated services.

However, developing a PCIU wouldn't be easy or cheap. Very rough estimates for providing a full range of services to a representative city of 90,000 people were: approximately 80 million computer instructions per second, 15,000 file accesses per second, 10 million statements of computer program, 7-10 years of development time and a development cost of \$500 million-\$1 billion. This puts the full-scale PCIU into the category of a major national effort; however, some perspective is restored when one considers that the nation's computing bill for the manned spaceflight effort during the 1960s has been estimated at \$2 billion. In any case, the PCIU certainly would need a great deal of careful preliminary planning before proceeding into development.

4. *The next steps toward a PCIU would be valuable whether or not they resulted in a PCIU.* These steps include:

- a. Development of a thorough, detailed PCIU plan including analysis of management, services, and technical alternatives, and delineation of principles and pitfalls common to any information utility implementation.
- b. Evaluation of related experiences to date, and

their implications with respect to PCIU development and operation.

- c. Development and evaluation of some low-cost pre-prototypes, involving groups of 100-1000 users and based on existing resources; e.g., an existing CATV system, an existing educational network, or a just-developing planned community.

Even if such studies were not followed by actual development of a PCIU, the insights they would provide on the social and economic implications of any sort of information utility would be invaluable in guiding the development of alternative systems, in order to make sure that the utility would serve the people and not vice versa.

Software Design for the Community Information Utility

by DONALD COHEN

The RAND Corporation
Santa Monica, California

The aim of our current efforts is to present a framework that will support further consideration, and hopefully development, of a Prototype Community Information Utility (PCIU). There is a continuing explosive growth in the number and size of data bases and information services that are potentially useful to various segments of the community. If a common point of contact can be developed among these data bases and services (no mean problem in itself), then the economic and social value of a properly integrated data bank could greatly exceed the value of the individual data bases. To exploit such a data bank in any effective manner requires a number of things:

- Data bases, individually and collectively, must be organized in a consistent manner, their utility and scope of application carefully defined, and their validity certified.
- A continuing process must be established for the collection, organization, evaluation, and certification of new data.
- The set of services that utilize the data bank must be carefully selected, defined, and implemented so that the system is not burdened with applications

that are individually of limited scope and difficult to interface with one another.

- The system that supports these services must be virtually failsafe, guarantee a very high degree of data security and privacy, and provide the basis for management mechanisms that insure its continued operation within accepted guidelines.

Preliminary estimates of the PCIU workload generated by a moderate sized city (75,000-100,000 people) indicate that an instruction execution rate of 80-85 million instructions per second and some 15-20 thousand file accesses per second will be required. Some 40-50 thousand on-line consoles will have to be supported with possibly 20-50 percent of these active at any one time. Extending the system to a large metropolitan area of several million potential users will require an exponential increase in system capacity and complexity.

No single processor exists currently, or is likely to exist in the foreseeable future, that can handle a workload of even the prototype system. A very large multi-processor system will be required in which groups of processors are devoted to one of the three major tasks of message concentration (front-end communications processors), message processing, and data management. Within each group processors may be either partially or completely interchangeable. In addition, redundant equipment will be required in case of severe hardware failures or system overloads. As the PCIU would be a large-scale social experiment, strong emphasis on system measurability is required.

The PCIU will be predominantly a closed system with its emphasis on well-defined applications that manipulate medium- to large-scale data bases. Development of new applications, on-line programming by users, and updates to various data bases will have to be rigidly controlled to insure that bread-and-butter applications receive sufficient support, to minimize the effects of system failures, and to maintain the integrity of the data bank.

The detailed design of the PCIU must be preceded by an in-depth analysis in several areas. First, potential applications must be examined to determine workload and file requirements, the needs of these applications in terms of system resources, and the interfaces among an application, the system, and other applications. This will provide the first realistic estimate of system size and complexity. Once this data is available, alternative software designs can be proposed and examined for the degree of difficulty inherent in each to provide the necessarily high level of overall system control.

A third stage of the pre-design process should be a simulation of feasible alternative designs including the mechanisms that are proposed to handle the high vol-

ume of processor and file activity that is anticipated. At the same time, those portions of the system design that are responsible for insuring that data can be kept secure and that the effects of system failure can be minimized should be subjected to a detailed evaluation. Only when there is sufficient confidence in the preliminary system design, along with quantitative justification for that confidence, should the detailed development of a PCIU be attempted.

Management Prospects and Problems

by BURT NANUS

University of Southern California
Los Angeles, California

There is no doubt that if a community information utility (CIU) is to be a reality, it will have to integrate smoothly with all other aspects of urban life. This represents an enormous challenge because it is clear that the CIU will produce fundamental and far reaching changes in the major subsystems of urban life—i.e., the economic, political, educational and life support subsystems—and it is by no means certain that all the changes will be beneficial.

The only way to realize the many benefits of the CIU is by the most scrupulous and careful management of its design, implementation and operation. The CIU must be managed so that resources are allocated for the effective achievement of social ends, and these ends must represent a balancing of the interests of at least three constituencies; as follows:

1. Society's Objectives—the CIU must be designed in such a way as to assure equal and fair treatment to all users, to contribute to individual self-fulfillment and to enhance the awareness and general welfare of the citizenry. To do this, it should place a higher priority on public than private services, should be self supporting in the long run, and should be operated so as to protect the privacy and dignity of individuals.
2. User Objectives—the users of the CIU should be able to expect reasonable and fair prices, high quality of service, protection for proprietary data and programs, and a voice in the setting of standards and priorities.

3. Supplier Objectives—suppliers of CIU equipment and services should receive fair profits, rewards for technical excellence and social concern, and protection from losses due to the actions of other suppliers, users or regulatory agencies.

Given an appropriate set of ends, of which the above is merely suggestive, it should be possible to explore the appropriateness of alternative organizational configurations. Many models are worth considering, and they cover the entire spectrum from a purely public agency to a privately held corporation. Along this spectrum are such models as urban entrepreneurship, the non-profit corporation, the heavily regulated but privately owned utility, the COMSAT-like consortium of private companies and the government regulatory agency model, to name a few. One model that appears particularly attractive at this time is the public authority form (e.g., the Port of New York Authority) which would permit the CIU to operate outside the regular structure of government with relative administrative autonomy, but within carefully defined limits and a mandate to act in the public interest.

The legal organization form of the CIU and the definition of the locus of power for policy making are only the first of a series of management problems that will have to be resolved in establishing the CIU. Other difficult problems are related to determination of acceptable levels of service, pricing structure, competitive structure, funding methods, government relationships, research and development policy, regulatory issues, consumer safeguards and public relations policies. Some aspects of these problems have been solved successfully in other contexts, but many are unique to the CIU and will require extensive experimentation and management research.

Planning Community Information Utilities

by NORMAN R. NIELSEN

Stanford University
Stanford, California

Other members of the panel have discussed the various services which a Community Information Utility (CIU) might provide as well as the various hardware, software, and communication facilities which

it might employ. The CIU's actual configuration and mix of services will be determined by a number of inter-related factors stemming from areas such as sociology, psychology, computer science, economics, political science, communications, and electrical engineering. Nevertheless, a study of the economic considerations which underlie the CIU concept can indicate the more likely paths for development.

What a CIU might "look like" is of major interest. Economic considerations point toward a CIU built around a cable system which would link terminal and computer for both input and output purposes. The home TV set would be the primary output device, with some slow speed (pointer, touch tone pad, keyboard, etc.) mechanism for user input. The "central" computer system of the CIU would likely be merely a message switching computer. It would pass inputs and outputs among and between the users on the cable and the various application or service computing systems as well as control the output video generation for cable users. The application systems would be developed and operated independently, although each would communicate with the central computer via a standard interface.

Although inter-CIU communication could be handled indirectly through each of the application services as appropriate, there are economic advantages to the direct connection of CIUs. The use of a network interface processor in conjunction with the central CIU computer would not only minimize communication resource usage but would also permit efficient use of services available on other CIUs. This latter situation has positive implications for development costs, start-up costs, and the required critical mass of a CIU (see below).

Despite the desirability, it is quite unlikely that the CIU will offer the full range of services that are frequently talked about. The provision of dynamic video output (e.g. film clips) to individual users is virtually prohibited by economic considerations. Voting services face very tough cost-benefit questions. On the other hand, some services such as education appear in a much more favorable light. Despite the independence of the various applications services, it would appear that the whole will be greater than the sum of the parts; that is, the individual applications will tend to become more valuable as additional applications are added to the CIU. The CIU will also face a critical mass effect in that a certain level and variety of service must be provided and used before the CIU can develop in a viable fashion.

The decentralized organization of the CIU is likely to imply a decentralized file structure, even though each service could talk to any other service. Such a development favorably affects the privacy and file

integrity issues, since it lessens protection problems and renders file integration more difficult. By the same token, however, it hinders the applications, efficiencies, and other benefits that would be realizable with integrated files.

Another major concern is the likely cost of a CIU. It would appear that the communication and terminal systems might run on the order of \$25 per user per month. Estimated usage costs (admittedly *very* gross) could easily run to another \$25 per terminal per month. Thus, the widespread use of CIUs could have a big impact upon consumption patterns as well as upon the manner in which many businesses are conducted.

This raises the question of how to pay for a CIU. It is most likely that there would be some combination of fixed charges and variable charges based upon actual usage. Many services would likely be subsidized or otherwise supported by the provider (rather than by the user). The computer base for the system opens up a number of possibilities for splitting payments between users, service operators, program developers, and other support organizations. The basic economics permit a wide range of alternatives, so it is likely that non-economic considerations will have a large impact upon the final charge structure.

Two economic problems face the development of a prototype CIU system. First, and most obvious, is the need for massive funding for software development and start-up costs. The second problem relates to the impact of the provision of these funds. Clearly one can't have users without services nor services without users. Hence, some type of subsidization or guarantee will likely be needed to get the prototype started. However, such support will alter participant behavior, partially obscuring the desired marketing and behavioral data. A number of other economic factors combine to indicate that the development and operation of a prototype CIU will be a valuable but non-straightforward endeavor.

Information Services

by EDWIN B. PARKER

Stanford University
Stanford, California

Two general classes of services will be required in a community information utility developed as an exten-

sion of cable television. One class is that provided by the private sector of the economy and the other is public sector services. With respect to all services available through the private sector, such as banking, shopping, entertainment, and all business and commercial services, the information utility should provide a standard information transmission service such that all potential suppliers of service can reach their potential customers. In other words, the utility should provide non-discriminatory competitive access to all computer and other information services without putting itself in the position of being a monopoly supplier of services. This implies that the utility specify technical interface, access and communication standards, but avoid responsibility for the contents of information transmitted through the utility. Detailed discussion with potential

suppliers of services will be required to establish adequate interface standards.

Special arrangements may have to be made to develop public sector services, the most important of which are education and information retrieval services. In the initial stages of development of education services, highest priority should be given to the delivery to homes of pre-school, supplementary and continuing education services. The greatest potential of the information utility may lie in its promise to provide economical and effective life-long learning. Also important will be the provision of public access via the utility to "public" government information that's otherwise difficult to obtain. Online voting and polling services should be given low priority or deferred because of a variety of political dangers.

A panel session—Computers and the problems of society

Computers and Urban Problems

by PETER KAMNITZER

University of California
Los Angeles, California

The recent focus of interest on urban problem solutions has progressed from early enthusiasm to the realization of the enormous difficulties awaiting the problem solver. The social and political problems attending problem perception and definition, priority establishment, cost and benefit distribution, information and program control seem to overwhelm the potentially available technological solutions to the manifestations of present urban ills.

Utilization of computer technology in the area of urban problems has grown from extensive data storage and retrieval and data analysis to simulation and modelling on a useful but as yet limited operational scale. Large scale simulation models have been attempted particularly with regard to urban transportation and its impact on land use. Computer graphics and on-line interactive man-machine systems are showing promise as useful aids to planning and decision making. Continuing progress will depend on further urban research; on development of uniform data formats; on faster, cheaper, more reliable and more powerful computers; and on financial and institutional encouragement.

Urban problem patterns on a short range scale will basically follow present physical and social trends resulting in further congestion, pollution, slums, sprawl, etc. On a long range scale they will increasingly be associated with the impact of technological and social forces on changing urban patterns within established as well as totally new concepts of urban life.

Computers can contribute to the amelioration of urban problems predominantly in two major categories: through their effect on a changing urban fabric and through their effect on the process of planning and decision making. They can be used in city building and rebuilding through the utilization of automation, communication and systems control (automated transportation, construction methods, controlled environments,

interactive communication, etc.). Planning and decision making can be greatly enhanced by on-line interactive computer methods. The complex, open-ended urban system with its absence of clear goal definitions tends to defy total optimization. In contrast, the man-machine mode permits man's value judgments to become part of the problem solving process itself. An Urban Simulation Laboratory is proposed which would bring together all means of simulation (mathematical modelling, man gaming and perceptual environment simulation) for purposes of research and experimentation with hypothetical solutions to urban problems. Interactive information display would permit queries by researchers and community representatives in an "if-then" mode, and thus would significantly contribute to urban decision making within the context of an informed and participatory society.

Successful implementation of innovation depends on a general climate of positivism, government subsidy of innovation, retraining programs, continuing education, as well as on user participation in the planning and implementation processes. New institutional arrangements between universities, research institutions, government and private industry are suggested to maximize learning, research and problem solving opportunities. The author cautions not to forget the "art" of problem solving in the commendable attempt of creating a rigorously applied "science" for "Computers and the Problems of Society."

The Current Crisis in American Education

by NORTON F. KRISTY

Refocus
Los Angeles, California

Public education in America is in crisis. Its critics point out that it is not doing its job at all effectively—at least according to the current expectations of upgrading the economically and culturally handicapped. The costs of public education have risen alarmingly in the past

20 years, and are currently out-running the tax base at all levels of education from primary school to graduate school. In the early sixties, many educators joined forces with systems specialists and computer development people in what has turned out to be a romantic dream. The dream was to some way, some how, combine concepts of system analysis and computer technology with principles of programmed learning in a way that would "revolutionize learning". Those high hopes have proved to be remarkably short-lived.

Now, in 1971, the conservatives appear to be in ascendance. There is widespread disillusionment with the "failure" of educational technology and innovation. However, the failure of educational technology and computer applications has largely been the result of

1. Poor administration of research and development monies by a tangled skein of governmental agencies competitively involved in educational research.
2. A grossly inadequate funding program for educational research. Those monies which were available tended to be spent on short-term, fragmented research programs. In other words, there is an almost desperate need to coordinate research, particularly at the Federal level, into one reasonably well-orchestrated program that will support risk-taking, and will give a financial base to promising ideas for an extended period of time.
3. Premature implementation of educational technology. Implementation of computer applications to education must be at least as well planned as the implementation of a major new military system.

Since we in the educational community are now at a point of considerable disillusionment concerning the value of computer technology, the role of the Federal Government in the past five years needs to be recounted. The Bureau of Research of the U.S. Office of Education in 1966/67, planned for an accelerating program of investment in computer applications ranging from administrative applications to computer assisted instruction. Over the time period 1967-69, less than 20 percent of these planned-for funds were ever in fact expended. Many programs were initiated and then not funded. Those that were funded were given much shorter periods of time than had been initially planned to cover the proposed work.

In spite of this, considerable technical progress has been made on the effective application of computers in education. However, it is the political situation which

will control the future of wide-scale research, development and installation of such applications. The educational enterprise in America is a fragmented instrumentality composed of more than twenty thousand school districts and almost three thousand institutions of higher learning. This educational enterprise has no centralized authority that can promote, support and press for change. At the same time, it has most of the limitations of small-scale organizations without much compensating freedom of action or flexibility of response to user requirements. Finally, education in America has now run out of money. It cannot mount a sustained program of experimentation, development, and implementation dealing with the very technologies and instructional practices that could redeem it.

A highly feasible method of bringing computer technology to important use in public education in the next decade is a reorganized state/federal program of major proportion. This program, if it were to be created, should concentrate first on higher education, for the cost/effectiveness is greater there.

International Implications: Need for World Simulation

by JOHN McLEOD

SCi World Simulation

Technological trends are causing profound changes at all levels and in all sectors of society. Some of these changes are considered desirable by those affected, some are not.

Computers are so inseparably intertwined with technology that many of the undesirable, even alarming, trends are being blamed on computers. Whether or not this is justified, it seems that if the undesirable trends are to be checked and the desirable ones reinforced, we will have to call on computers for help.

The reason computers will be necessary is that problems of society today, stemming from or aggravated by the "population explosion" and its multiple side-effects, are much too complicated for comprehension by the unaided human intellect.

In the last analysis, if humanity is to survive, *people* must solve the problems of society. But first there must be *understanding*. And to acquire understanding, people must have a tool for keeping track of the myriad pieces

of information, and the dynamic interactions among them, that contribute to the problems and which must be taken into account in any proposed solution. If the interrelationships as well as the facts are properly fed into a computer, the result will be a computer model of the system of interest. Experiments can then be designed and run on the model which will impart an understanding of the real-world situation.

However, all sub-systems of our society are so inter-related, even up to and including nations, that only a model including all the nations of the world can give us insight into social problems which transcend national boundaries—as so many important ones do.

For the foregoing reasons it is urged that work on the development of a world simulation be officially encouraged and adequately funded by our government.

Computers and National Security

by E. W. PAXSON

The RAND Corporation
Santa Monica, California

The digital computer was spawned by World War II. Military requirements have continued to pace computer development. Computer technology and weapon system sophistication have marched in tandem. Neither has dominated, but current weapon systems, operations, and management are impossible without the computer. Almost 90 percent of the Government's current stock of 5000 plus computers are devoted to defense and space activities, reversing the civilian use pattern of at least ten times that number of machines.

Weapons have been developed and fielded in response to a real external threat. But there has been no true arms race and one is unlikely. We have relied largely on advanced technology to generate a posture deterring the catastrophe of nuclear war. Our lead in computer science is a major contributing factor in giving us the technological edge and in contributing to deterrence.

Will we continue to have this edge? Pressures from the domestic sector, the flyback effect after termination of our involvement in South East Asia, our hopes for favorable Strategic Arms Limitations Talks will undoubtedly lead to a decrease in defense funding. Since our Research and Development system, unlike that of Russia, is closely coupled to weapon system develop-

ment, there will be a cutback in R&D, including computer sciences, as new systems are cancelled or stretched out.

Scientific research has relied heavily on Government funding. Current Congressional attitudes toward basic research are that it must be directly 'relevant' to military matters. In the USSR, the State has opposite attitudes.

Are industrial motivations strong enough to fill these gaps?

Under the concept of strategic sufficiency, President Nixon has asked for options of greater flexibility in deterrence and nuclear war management than the implementation of Assured Destruction which can imply the death of half of our people in retaliation.

The computer implications are heavy. Surveillance and all operating weapon systems must be tightly linked to produce the required data base update to permit finger-tip command and control of a major crisis. Not only are the data processing requirements immense, but there is imperative need for adaptive, on-line, man-machine intelligence—not artificial intelligence, to explore the 'what if?' 'what then?' of combat situations in much less than all too short real time. We still talk to computers and not *with* them. Machine technology on the LSI and memory levels is well out of balance with required and expensive software.

As military budgets are decreased, basic research and development should obviously increase in proportion. But, as usual, I think it will take its share of a slash. I hope I am wrong.

Ecological Problems

by ROGER WEINBERG

Kansas State University
Manhattan, Kansas

Man has exploited nature during the 19th and 20th centuries. Therefore, in America he has changed many productive Indian systems of life: the Oklahoma plains of the Kiow, rich in grass and buffalo, into a dust bowl; the Washington salmon streams of the Haida into a sequence of DDT-poisoned reservoirs; the British Columbian Kootenay Lake of the Tlinglit, filled with fish, into a recipient for fertilizers.

By 1971 he had gone further, filling the atmosphere

with carbon monoxide and other noxious gases so that breathing in New York City was equivalent to smoking a pack and a half of cigarettes a day. He has even polluted the vast life-giving ocean with death-dealing poisons. Mercury contaminated the swordfish which became dangerous for humans to eat. DDT slowed the ocean plants' ability to capture the energy of the sun in the vital first step of a long food chain leading to man.

As man was developing a technology which created these problems, he was with the same technology, developing means for solving them. By 1946 he had built thinking machines—electronic computers, and by 1971 he had used them to model eco-systems, to optimize the results of resource management, and to coordinate research efforts by teams of individual research workers who were scattered by distance.

Future developments such as small, cheap minicomputers can provide computers for gathering weather data at remote Pacific island stations, while powerful parallel processing computers will be capable of running models of large complex weather systems.

Along with new computers, a new technique such as microprogramming will enable a programmer to set up computer circuits tailored for his particular program, and a new concept such as the computer utility will

enable groups of programmers to communicate with each other, and to utilize the power of a large central computer.

These new computers, techniques, and concepts are man's genie. And man, become Aladdin, will be able to:

- a. Improve weather forecasting, and be forewarned against natural disasters.
- b. Plan the optimal use of scarce natural resources.
- c. Simulate to improve decision making, testing alternate ecological policies in order to choose the best one before it is implemented, thereby avoiding dangerous mistakes.
- d. Plan and coordinate measures in pollution control.
- e. Build information retrieval systems which make scientific and technical data accessible to interdisciplinary teams studying world-wide environmental systems.

Computers provide man with the power of vision into alternate future worlds, and the option of choice among these worlds. What choice he makes is his decision. Whatever his choice, he will live in the heaven he creates, or in the hell.

AMERICAN FEDERATION OF INFORMATION PROCESSING SOCIETIES, INC. (AFIPS)

AFIPS OFFICERS and BOARD OF DIRECTORS

President

Mr. Keith Uncapher
The RAND Corporation
1700 Main Street
Santa Monica, California 90406

Vice President

Mr. Walter L. Anderson
General Kinetics, Inc.
11425 Isaac Newton Square, South
Reston, Virginia 22070

Secretary

Dr. Donald Walker
Artificial Intelligence Group
Stanford Research Institute
Menlo Park, California 94025

Treasurer

Dr. Robert W. Rector
University of California
6115 Mathematical Sciences Building
Los Angeles, California 90024

Executive Director

Dr. Bruce Gilchrist
AFIPS
210 Summit Avenue
Montvale, New Jersey 07645

ACM Directors

Mr. Walter Carlson
IBM Corporation
Armonk, New York, 10504

Mr. Donn B. Parker
Stanford Research Institute
Menlo Park, California 94025

Dr. Ward Sangren
The University of California
521 University Hall
2200 University Avenue
Berkeley, California 94720

IEEE Directors

Mr. L. C. Hobbs
Hobbs Associates, Inc.
P.O. Box 686
Corona del Mar, California 92625

Dr. Robert A. Kudlich
Raytheon Co., Equipment Division
Wayland Laboratory
Boston Post Road
Wayland, Massachusetts 01778

Professor Edward J. McCluskey
Stanford University
Department of Electrical Engineering
Palo Alto, California 94305

Simulations Council Director

Mr. James E. Wolle
General Electric Company (VFSTC)
Space Division
P.O. Box 8555
Philadelphia, Pa. 19101

*American Institute of Aeronautics and
Astronautics Director*

Dr. Eugene Levin
Culler-Harrison Company
5770 Thornwood Drive
Goleta, California 93017

American Statistical Association Director

Dr. Martin Schatzoff
IBM Cambridge Scientific Center
545 Technology Square
Cambridge, Massachusetts 02130

Instrument Society of America Director

Mr. Theodore J. Williams
Purdue Laboratory for Applied Industrial Control
Purdue University
Lafayette, Indiana 47907

Society for Information Display Director

Mr. William Bethke
RADC (EME, W. Bethke)
Griffis Air Force Base
Rome, New York 13440

Association for Computational Linguistics Director

Dr. A. Hood Roberts
Center for Applied Linguistics
1717 Massachusetts Avenue, N.W.
Washington, D.C. 20036

*American Institute of Certified Public
Accountants Director*

Mr. Noel Zakin
Computer Technical Services
ACIPA—666 Fifth Avenue
New York, New York 10019

American Society for Information Science Director

Mr. Herbert Koller
ASIS
1140 Connecticut Avenue, N.W. Suite 804
Washington, D.C. 20036

Society for Industrial and Applied Mathematics Director

Dr. D. L. Thomsen, Jr.
IBM Corporation
Armonk, New York 10504

Special Libraries Association Director

Mr. Burton E. Lamkin
Office of Education—Room 5901
7th and D Streets, S.W.
Washington, D.C. 20202

JOINT COMPUTER CONFERENCE BOARD

President

Mr. Keith W. Uncapher
The RAND Corporation
1700 Main Street
Santa Monica, California 90406

Vice President

Mr. Walter L. Anderson
General Kinetics, Incorporated
11425 Isaac Newton Square, South
Reston, Virginia 22070

Treasurer

Dr. Robert W. Rector
University of California
6115 Mathematical Sciences Building
Los Angeles, California 90024

ACM Representative

Mr. Richard B. Blue Sr.
1320 Victoria Avenue
Los Angeles, California 90019

IEEE Representative

Dr. Robert A. Kudlich
Raytheon Company, Equipment Division
Wayland Laboratory
Boston Post Road
Wayland, Massachusetts 01778

SCI Representative

Mr. John E. Sherman
Lockheed Missiles and Space Co
Org. 19-30, Building 102,
P.O. Box 504
Sunnyvale, California 94088

JOINT COMPUTER CONFERENCE
COMMITTEE

Dr. A. S. Hoagland, Chairman
IBM Research Center
P.O. Box 218
Yorktown Heights, New York 10508

JOINT COMPUTER CONFERENCE TECHNICAL
PROGRAM COMMITTEE

Mr. David R. Brown, Chairman
Stanford Research Institute
333 Ravenswood Avenue
Menlo Park, California 94025

FUTURE JCC CONFERENCE CHAIRMEN

1972 SJCC

Mr. Jack E. Bertram
IBM Corporation
P.O. Box 37
Armonk, New York 10504

1972 FJCC

Dr. Robert Spinrad
Xerox Data Systems
701 South Aviation Blvd.
El Segundo, California 90245

1971 FJCC STEERING COMMITTEE

General Chairman

Ralph R. Wheeler
Lockheed Missiles and Space Company

Vice Chairman

Albert C. Porter
California Public Utilities Commission

Secretary

Joseph M. Crosslin
Control Data Corporation

Treasurer

Corydon Hurtado
Cyberrary International Company

Technical Program

Dr. Martin Y. Silberberg—Chairman
IBM Corporation
Robert Blumenthal—Vice Chairman
IBM Corporation

Local Arrangements

Thomas Bieg—Chairman
IBM Corporation
Kenneth W. Charshaf—Vice Chairman
Bank of America

Registration

Robert Borkenhagen—Chairman
ISI Corporation
Gary Thomasson—Vice Chairman
Trans-A-File Systems Company

Printing and Mailing

Jeffrey D. Stein—Chairman
On-Line Business Systems, Inc.
Eckart Sellinger—Vice Chairman
Bank of America

Exhibits

Jack Miller—Chairman
Ampex Corporation
Clyde Cornwell—Vice Chairman
Ampex Computer Products Division

Special Activities

Norman Kristovich—Chairman
Dept. of Industrial Relations
David Wilkinson—Vice Chairman
Hewlett Packard International Corp.

Public Relations

Frederick M. Hoar—Chairman
Fairchild Camera & Instrument Corp.
Ronald R. Batiste—Vice Chairman
System Development Corporation

ACM Representative

Thomas E. Murray
Del Monte Corporation

IEEE Computer Society Representative

Terry Ruster
Fairchild Corporation

SCI Representative

J. E. Sherman
Lockheed Missiles and Space Company

JCC Committee Liaison

Dr. Morton M. Astrahan
IBM Corporation

REVIEWERS, PANELISTS, AND SESSION CHAIRMEN

SESSION CHAIRMEN

Baran, Paul
Bell, C. Gordon
Bennett, John L.
Blois, Marsden S.
Borko, Harold
Coate, Robert E.
Farber, Dave
Frederickson, A. Anton, Jr.
Gould, Kent
Hamming, Richard
Haynes, Herb
Hisey, Bradner L.

Hoffman, Lance J.
Howard, John
King, Warren
Kuney, Joseph
La Riviere, Dave
Lipton, Harry
Madden, John D.
Mason, Maughan S.
Newton, Carol
Nigh, Max T.
Nilsen, Raymond N.

Ponder, Leonard H.
Purdy, Gerry
Ross, L. W.
Sackman, H.
Schroeder, David L.
Schwetman, Herb
Warlick, Charles
Weiss, Donald H.
Wilkinson, Dave
Williams, Robert
Wormeli, Paul

PANELISTS

Barg, Benjamin
Bekey, G. A.
Blease, Thomas
Boudreau, P. E.
Brandt, Gil
Brooks, F. P.
Burke, Robert
Caine, S. H.
Chien, R. T.
Cohen, N. D.
Courtney, Robert
Cserhalmi, N.
Dalkey, Norman
Davis, Dan
Edwards, D. B. G.
Engelbart, D. C.
Epple, Ken
Everett, R. R.
Foster, J. E.
Frank, A. A.
Frazer, J. W.
Freeman, R. B.
Griswold, R. E.

Harding, P. A.
Hawley, C. L.
Hugo, F. M.
Jeffries, S. B.
Kamnitzer, P.
Karplus, W. J.
Katter, R. V.
Kay, A. C.
Korn, G. A.
Kristy, N. F.
Lalchandani, A.
Lamson, B.
Larson, Dave
Levinthal, C.
McLeod, J.
McClure, R. M.
-aginniss, F. J.
Maley, G. A.
Mallender, Ian
Martin, D. C.
Merritt, M.
Mitchell, Kent
Morris, J. B., Jr.

Morton, M. S.
Morton, N. E.
Nanus, B.
Nathan, R.
Nielson, N. R.
Norberg, G. R.
Parker, E. B.
Paxson, E. W.
Pollard, B. W.
Post, C.
Raub, W.
Rosen, Saul
Ryan, Frank
Sibley, E. H.
Smith, C. L.
Tatum, Liston
Van Brunt, E. E.
Walker, D. E.
Weinberg, R.
Weissman, Clark
Yamamoto, W.
Yarrington, A.

REVIEWERS

Abbott, Robert P.
Acton, Forman S.
Adams, Edward N.
Aiken, Robert M.
Alcorn, Bruce K.
Allen, Roy P.
Amarel, Saul
Anderson, James P.
Anderson, Robert H.
Anderson, Thomas C.
Anzelmo, Frank

Arndt, Fred R.
Arnovick, George N.
Axsom, Larry E.
Badger, George F., Jr.
Ball, N. Addison
Barcelo, Wayne R.
Barlow, Allen E.
Barnes, Ben B.
Barnett, Robert M.
Bartlett, James P.
Bayles, Richard U.

Belady, L. A.
Bell, Thomas E.
Berglass, Gilbert R.
Berning, Paul T.
Bethke, William P.
Beyer, William A.
Black, Donald V.
Bodoia, Morris J.
Bolton, Gordon R.
Borko, Harold
Bratman, Harvey

Bredt, Thomas H.
Bremer, John W.
Brennan, Robert D.
Brown, Ralph R.
Browne, J. C.
Bryan, G. Edward
Burkhard, Walter A.
Caine, Stephen H.
Calhoun, Kenneth J.
Calhoun, Myron A.
Calingaert, Peter
Calvert, Thomas W.
Canaday, R. H.
Cardwell, David W.
Carmon, James L.
Chaitin, Leonard J.
Chandler, John P.
Chapman, R. G., Jr.
Cheydleur, Benjamin F.
Chow, W. F.
Clymer, A. Ben
Cocanower, Alfred B.
Coles, L. Stephen
Collmeyer, Arthur J.
Condon, S. F.
Connors, Michael M.
Constant, Robert N.
Cook, Jeffrey D.
Cooke, Walter F.
Corduan, Alfred E.
Corwin, Barnet C.
Cotton, Ira W.
Coulman, George A.
Cowan, Robert
Critchlow, Arthur J.
Critchlow, Dale L.
Cserhalmi, Nicholas
Csuri, Charles A.
Curtis, Kent K.
Dale, A. G.
Daniel, Walter E., Jr.
Darms, Donald A.
DeJong, S. Peter
Denes, John E.
Denning, Peter J.
Deveber, Jeffrey L.
Dimmler, D. Gerd
Dodd, George G.
Douglas, John R.
Dove, Richard K.
Duffendack, John C.
Duggan, Michael A.
Dumey, Arnold I.
Earnest, Lester D.
Eisenstark, Raymond
Ellin, Everett

Estes, Samuel E.
Farmer, Nick A.
Feurzeig, Wallace
Feustel, Edward A.
Firschein, Oscar
Flanagan, J. L.
Foster, John E.
Fox, Margaret R.
Frank, Amalie J.
Fraser, A. G.
Futterweit, Adolf
Gardner, Reed M.
Geyer, James B.
Gilliand, B. E.
Goetz, Martin A.
Gold, Michael M.
Gotterer, Malcolm H.
Greenawalt, Eddie M.
Greenfield, Martin N.
Haberman, Eugene J.
Haibt, Luther H.
Haims, Murray J.
Hammer, Carl
Haney, Frederick M.
Hanlon, A. G.
Hansard, Robert M.
Harding, Philip A.
Harrison, Joseph O., Sr.
Hartwick, R. Dean
Hathaway, Allen W.
Hedrick, George Ellwood, III
Heilweil, Melvin F.
Hermann, Paul J.
Herzog, Bertram
Hinrichs, Joe
Hodes, Louis
Hollander, Gerhard L.
Hodper, Robert L.
Humphrey, Thomas A.
Hyatt, Gilbert P.
Jameson, Wm. J., Jr.
Jeffries, Ronald E.
Jessep, Donald C.
Kain, Richard Y.
Kaitz, Marvin J.
Kalin, Richard B.
Keenan, Thomas A.
Keller, Roy F.
Kahalil, Hatem M.
King, Robert E.
King, Willis K.
Kinney, Edward S.
Klerer, Melvin
Klir, George J.
Knupp, John L., Jr.
Koen, Henry R., Jr.

Koller, Herbert R.
Koory, Jerry L.
Kopf, John O.
Kovach, Ladis D.
Kurtz, Thomas E.
Lambert, Robert J.
Lampson, Butler
Landoll, James R.
Larkin, Robert C.
Leathrum, James F.
Lenahan, John J.
Lett, A. S.
Lewis, William E.
Lindahl, Charles E.
Lindenmeyer, Leonard R.
Linville, Thomas P.
Liu, Ho-Nein
Livdahl, Richard C.
Lomet, David B.
Long, Henry A.
Mason, Maughan S.
McClure, Robert M.
McCoy, Maurice E., Jr.
McFarland, Clay
McKnight, Randy S.
McLeod, John
Machover, Carl
Main, Walter
Malone, Charles M.
Marcotty, Michael
Meadows, H. E.
Miles, E. P., Jr.
Miller, William G.
Moe, Maynard L.
Morrison, James F.
Myers, Robert P.
Nanus, Burt
Nassir, Andrew M.
Neilsen, Norman R.
Notz, William A.
O'Brien, Joseph A.
Paden, Douglas R.
Page, Carl Victor
Parker, Donn B.
Passaretti, Anthony
Pattee, Harold E.
Pearson, Karl M., Jr.
Pomerance, Richard M.
Pounds, Kenneth
Pritchard, J. Paul, Jr.
Rahe, George A.
Ralston, Anthony
Ramamoorthy, C. V.
Remson, Irwin
Rigney, Joseph W.
Rubey, Raymond J.

Ruffing, Linus F.
Sanborn, Jere L.
Schafer, Ronald W.
Schischa, Eywin
Schwenker, J. E.
Sedelow, Sally Yeates
Seed, John C.
Sheldon, Robert C.
Shipman, Jerome S.
Shuey, Richard L.
Slaughter, Barbara G.
Slutz, Donald R.
Smith, Cecil L.
Springe, Fred W.

Starkweather, John A.
Stewart, David H.
Stewart, Robert M.
Sturm, Walter A.
Summit, Roger K.
Tan, Chung-Jen
Uber, Gordon T.
Van Brink, Herbert F.
Van Tassel, Dennie
Vemuri, V.
Vichnevetsky, Robert
Wadia, Aspi B.
Wait, John V.

Walker, P. Duane
Wallace, John B., Jr.
Warheit, I. A.
Weissman, Clark
Wigington, Ronald L.
Wilborn, R. C.
Wilcox, Lyle C.
Wilkov, Robert S.
Willard, Donald A.
Williams, Theodore J.
Wolle, James E.
Wyman, John C.
Yau, Stephen S.

PRELIMINARY LIST OF EXHIBITORS

Addison-Wesley Publishing Company
Addressograph Multigraph Corp.
AFIPS Press
American Telephone & Telegraph Co.
Ampex Corporation
Applied Magnetics Corp. (with Standard Memories)
Auerbach Info, Inc.
Auricord Div.—Scovill Mfg. Co.
Automata Corporation
Bell & Howell, E & IG
The Bendix Corporation
Benwill Publishing Corp.
Boeing Computer Services, Inc.
Bryant Computer Products
Bucode
Bunker Ramo
Caelus Memories
California Computer Products, Inc.
Cambridge Memories, Inc.
Canada: Dept. of Industry, Trade & Commerce
Canberra Industries
Centronics Data Computer Corp.
Century Data Systems, Inc.
Cincinnati Milacron
Cipher Data Products, Inc.
Clasco Systems, Inc.
Codex Corp.
Collins Radio Company
ComData Corporation
Com-Mark, Inc.
Compucorp (A Div. of Computer Design)
Computer Automation, Inc.
Computer Communications, Inc.
Computer Decisions
Computer Design Publishing Corp.
Computer Intelligence Corp.
Computer Investors Group, Inc.
Computer Terminal Corp.
Computer Transceiver Systems, Inc.
Computerworld
Control Devices Inc.
Courier Terminal Systems, Inc.
Cybercom Corporation
Data Disc, Inc.
Data General Corporation
Datamation
Data Printer Corp.
Datapro Research Corporation
Data Products Corporation
Dataram
Datawest Corporation
Diablo Systems Inc.
A. B. Dick Company
Dicom Industries
Digi-Data Corporation
Digital Computer Controls
Digital Development Corp.
Digital Equipment Corporation
Digitronics Corporation
Eastman Kodak Company
E-H Research Laboratories, Inc.
Electronic News—Fairchild Publications
Electronic Processors, Inc.
Fabri-Tek, Inc. Memory Products Div.
Facit-Odhner, Inc.
Gould Inc., Brush Div.
Grumman Data Systems Corp.
GTE Information Systems Inc.
GTE Lenkurt Inc.
GTE Sylvania
Hewlett-Packard
Hitchcock Publishing
Houston Instrument
IEEE Computer Society
Incoterm Corporation
Inforex, Inc.
Information Control Corporation
Input Output Computer Services, Inc.
Instronics Limited
Interdata, Inc.
International Data Corp.
International Teleprinter Corp.
I/O Devices, Inc.
Intel Corp., Information Storage Systems Div.
Kanematsu-Gosho (U.S.A.) Inc.
Kennedy Company
Keuffel & Esser Company
Kybe Corporation
Licon Div. I.T.W.
Lipps., Inc.
Litton ABS OEM Products
Litton Industries

Lorain Products Corp.
Lundy Electronics & Systems Inc.
3M Company Instrument & Data Products
Magnusonic Devices, Inc.
Marshall Data Systems
Memory Systems, Inc.
Microdata Corporation
Micro Switch, A Div. of Honeywell
Milgo Electronic Corporation (ICC)
Miratel Div.—BBRC
Modern Data Services, Inc.
Mohawk Data Sciences Corp.
Nashua Corporation
NCR
Nortronics Company, Inc.
Numeridex Tape Systems, Inc.
Optical Business Machines, Inc.
Optical Scanning Corporation
Pacific Micronetics, Inc.
Panasonic
Paradyne Corporation
Penril Data Communications, Inc.
Peripheral Data Machines, Inc.
Peripheral Equipment Corporation
Phonocopy, Inc.
Pioneer Magnetics, Inc.
Potter Instrument Company, Inc.
Precision Instrument
Prentice Hall, Inc.
Princeton Electronic Products, Inc.
Quadri Corporation
Raytheon Company
Remex, A Unit of Ex-Cell-O Corp.

Sangamo Electric Company
Signal Galaxies, Inc.
The Singer Company (Librascope Div.)
Singer—Micrographics Systems
Sola Electric
Sorbus, Inc.
Spartan Books
Storage Technology Corporation
The Superior Electric Company
Sycor, Inc.
Sykes Datatronics, Inc.
Tally Corp.
Tandberg of America, Inc.
Tec, Incorporated
Techtran Industries, Inc.
Tektronix, Inc.
Teletype Corp.
Telex/Communications Div.
Thomson-CSF Electron Tubes, Inc.
Timeplex, Inc.
Time Share Peripherals Corp.
Tracor Data Systems
United Telecontrol Electronics, Inc.
Unicomp, Inc.
Van San Corporation
Varian Data Systems
Video Systems Corp.
Wang Computer Products, Inc.
Warner Electric
Western Union Data Services Co.
Western Union Telegraph Company
John Wiley & Sons, Inc.
Xerox Corporation—Xerox Data Systems

AUTHOR INDEX

- Adams, M. C., 477
Adelman, A. G., 455
Amiot, L., 31
Aschenbrenner, R. A., 31
Asman, E. Z., 233
Aus, H. M., 379
Austin, J. E., 541
Baca, R. L., 309
Barney, G. O., 631
Bateman, B. L., 89
Bekey, G. A., 401
Bell, C. G., 387
Bennett, J. L., 197
Berg, R. O., 177
Berman, R. A., 369
Boehm, B. W., 669
Boehm, S. C., 309
Boudreau, P. E., 9
Brandt, G., 397
Bravdica, S. A., 225
Brooks, F. B., Jr., 395
Carroll, J. M., 571
Chamberlin, D. D., 263
Chambers, M. G., 309
Chappell, S. G., 651
Chew, P., 233
Clark, R. L., 369
Cleveland, W. B., 213
Cohen, N. D., 670
Coleman, N. L., 65
Covvey, H. D. J., 455
Crawford, P. B., 89
Deland, E. C., 369
Drew, D. D., 89
Edwards, D. B. G., 395
Elliott, W. D., 533
Eppele, L., 397
Epstein, G., 663
Felderhof, C. H., 455
Forman, E. H., 51
Frank, A. A., 357
Frank, A. J., 135
Freeman, R. B., 1
Gack, G., 295
Gallati, R. R. J., 303
Gilmore, P. A., 411
Gottlieb, S. E., 603
Gracon, T. J., 549
Graves, G. W., 123
Groner, G. F., 369
Hansen, M. H., 579
Hansen, W. J., 523
Hinkelman, K. W., 65
Hirschsohn, I., 501
Hodges, J. D., Jr., 281
Hoehenwarter, W. P., 639
Hoffman, L. J., 587
Jackson, R. S., 225
Jen, T. S., 171
Kamman, A. B., 17
Kamnitzer, P., 675
Kay, A., 395
Kennicott, P. R., 423
Klopfenstein, C. E., 435
Kolechta, W. J., 65
Korn, G. A., 379
Kristy, N. F., 675
Laga, E., 477
Lalchandani, A., 398
Lamson, B. G., 195
Langlois, W. E., 97
Learman, I., 469
Levinthal, C., 199
Lifshin, E., 423
Loeber, N. C., 79
McHardy, L., 571
McLeod, J., 676
Maholick, A. W., 1
Martin, D. C., 361
Martin, R., 571
Medak, G. M., 295
Mendler, P., 455
Menninga, L. D., 145
Merrit, M. J., 351
Mesquita, A. L., 27
Mishelevich, D. J., 271
Mitchell, K., 399
Moravec, H., 571
Morey, R., 477
Morton, N. E., 199
Nakamura, G., 57
Nanus, B., 671
Natarajan, N. K., 31
Nathan, R., 200
Newbery, A. C. R., 419
Newell, A., 387
Nielsen, N. R., 671
Nolby, R. A., 549
O'Connor, D. G., 203
Olsen, D. J., 115
Parker, E. B., 672
Patterson, A. C., 575
Paxson, E. W., 677
Peck, P. L., 561

Pendleton, J. C., 491
Perone, S. P., 441
Pingry, D. E., 123
Post, C. T., Jr., 195
Potas, W. A., 533
Prerau, D. S., 153
Pringle, W. L., 309
Purdy, K. G., 399
Raub, W. F., 201
Reich, K. E., 639
Rodriguez, E. J., 71
Ross, L. W., 105
Ryan, F. B., 400
Sammet, J. E., 243
Sanford, J. E., 233
Sansom, F. J., 549
Scavullo, V. P., 423
Schumacker, B., 619
Sheridan, T. B., 327
Sicko, J. S., 423
Sinclair, R., 351

Smith, C. C., 609
Steen, R. F., 9
Strauss, J. C., 39
Tang, C. K., 163
Taylor, K. W., 455
Thurber, K. J., 177
Turoff, M., 317
Umpleby, S., 337
Ung, M. T., 401
Van Brundt, E. E., 196
Van Dam, A., 533
Wegbreit, B., 253
Weinberg, R., 677
Whinston, A., 123
Whisenand, P. M., 295
White, M. S., Jr., 609
Wigle, E. D., 455
Wilkins, C. L., 435
Wood, D. C., 51
Yamamoto, W. S., 201
