Contract
NAS9-18411
MCR-91-1393
October 1993

SINDA/FLUINT

Systems Improved Numerical Differencing Analyzer and Fluid Integrator

Version 2.6

USER'S MANUAL

FOREWORD

The Systems Integrated Numerical Differencing Analyzer – 1985 Version (formerly called SINDA '85) is an upgraded version of the SINDA program. This new version consists of an entirely new preprocessor and a processor library that is much simpler than its predecessor in terms of the number of network solution routines available to the user. Major improvements over older SINDA codes include submodels and a comprehensive simultaneous fluid flow analysis capability: FLUINT.

SINDA/FLUINT has been well received since its first introduction in 1985, having effective supplanted other SINDA versions, becoming a best–seller and a NASA Space Act Award winner in the process. Originally intended to satisfy the ever–changing modeling needs of the spacecraft and launch vehicle thermal community, the program has diffused into other industries such as nuclear, aircraft, HVAC, and automotive, and into other specialties and subsystems such as propulsion and environmental control.

SINDA is intended primarily for analyzing thermal systems represented in electrical analog, lumped parameter form, although its use may be extended to include other classes of physical systems which can be modeled in this form. Potential SINDA users who are unfamiliar with the technique of modeling thermal systems should consult standard texts on the subject* prior to undertaking a study of Sections 1 through 4.

Users of older versions of SINDA will notice a great deal of commonality with the predecessor program, as well as some changes that are far reaching in implication, such as submodels. In the areas of node, Q-source and conductor definition, commonality was preserved between SINDA and SINDA/FLUINT. These areas generally comprise the great majority of a model's input, so a large percentage of upward-compatibility exists. Other areas of minimal change are in the subroutine argument lists and the syntax of M-type logic statements.

The FLUINT capability was originally added to SINDA '85 under NASA JSC Contract NAS9-17448, "Generalized Fluid System Analysis with SINDA '85." Its capabilities continue to expand under the current NASA JSC contract NAS9–18411, "Maintenance and Enhancement of the SINDA '85/FLUINT Code."

FLUINT is an advanced one dimensional fluid analysis code that solves arbitrary fluid flow networks. The networks may contain single–phase vapor or liquid as the working fluid, or combinations of liquid and/or vapor at various locations in the network. In general, these networks are solved in conjunction with one or more accompanying thermal networks, thus allowing the energy flows in and out of the fluid system(s) to be accounted for. FLUINT replaced the SINFLO feature found in the predecessor SINDA program.

Gratifyingly, this SINDA has become the definitive SINDA code for thermal/fluid analysis. FLUINT, which started as an adjunct, now outweighs the remainder of the code. Therefore, some new naming conventions will be used. The cumbersome *'85* will be dropped where possible: *SINDA/FLUINT* will be used to refer to the whole program, especially when used to distinguish it from older thermal–only versions. *SINDA* will be used to refer to the thermal capabilities, and *FLUINT* to the fluid flow capabilities. *SINDA'85* will be used only where needed to distinguish new thermal analysis capabilities from those of older SINDAs.

---

* See for instance: Dusinberre, G.M., *Heat Transfer Calculations by Finite Differences*, International Textbook Co., Scranton, PA, 1961.

ACKNOWLEDGMENTS

B.A. Cullimore and S.G. Ring are the principal authors of SINDA/FLUINT. However, SINDA has been an evolutionary product that is now over three decades old, and improvements would not have be possible without NASA's sponsorship and continuing commitment. A number of people deserve a great deal of credit for their efforts in establishing SINDA as an industry standard in the thermal/fluid system analysis community. Acknowledgments are due to:

J.D. Gaski, who made the whole concept real, with the ancestral CINDA program.

S.G. Ring, R.G. Goble, R.J. Connor, R.E. Kannady and G.M. Holmstead, for contributions to the SINDA processor, some of which was taken from the MITAS program. Mr. Ring is also due credit for the complete renovation of the preprocessor.

J.T. Skladany and F.A. Costello, for their SINDA time step selection algorithm.

J.W. Orsag, technical monitor for the original SINDA '85 development.

B.A. Cullimore, for the conception and implementation of FLUINT.

C.H. Lin and E.K. Ungar, the technical monitors for FLUINT development.

M. Welch and S. Damico, for the difficult task of independent code verification for FLUINT and SINDA, respectively.

In many ways, the creators, authors, and sponsors of SINDA/FLUINT can no more take credit for its success than can the authors of Fortran. Ultimately, the creativity and perseverance of many clever users has been one of the most important ingredients. Fortunately (from the authors' perspective), they are too numerous to thank individually.

VERSION 2.6 vs. VERSION 2.5

As an aid to users of Version 2.5, the major differences between Version 2.6 and Version 2.5 are summarized below. Changes to the thermal analysis capabilities were minor. Furthermore, few changes had an impact on the user interface. Most of the work reflected improvements to internal solution methods as needed to provide more robust operation.

A minor change to code was required immediately following the last publication of the manual: the FLUINT path keyword "AFT" was globally changed to "AFTH" to avoid conflicts with existing NASA space shuttle models. This change, which constitutes the only difference between Version 2.5 and 2.5A, is reflected in this revision. AFTH is only used in choked flow modeling (Section 6.9.6.6).

Thermal Analysis Improvements

STEADY-STATE CONVERGENCE—Corrections have been made to the logic that controls convergence of multiple thermal submodels. This logic results in speed enhancements by placing semi–converged submodels in a dormant state (refer to the ITHOLD control constant).

NEW TIME STEP PREDICTOR—The previous thermal time step predictor for the FWDBCK solution routine has been completely replaced. While based on the same theory as before, the new routine reestablishes the user's ability to invoke backups, better tolerates uneven time steps, and better detects step changes in parameters and boundary conditions.

Fluid Analysis Improvements

INTERNAL TIME STEP BACK UP—The primary purpose of Version 2.6 was the addition of an internal back up feature to the FLUINT time step integration. Each transient solution step is now reviewed for signs of inaccuracies and instabilities, and is either accepted or rejected. If rejected, the integration is repeated with a smaller time step. This improvement is especially helpful in violent two–phase transients. Otherwise, the existence of this module is largely transparent to the user.

EXPANDED TOOLS FOR NONEQUILIBRIUM CONTROL VOLUMES—Two new options exist for the NONEQ simulation tools. The first option is a new routine that allows the analyst to reset certain control parameters that were previously hard–wired. The second option is the ability to better model continuous boil–off and condensation processes.

MISCELLANEOUS IMPROVEMENTS—(1) A new option has been added to 6000 series fluids that helps avoid discontinuities in the P–V–T surface in the cold supercritical regime; (2) K–factors have been enabled for slip flow paths; (3) potential energy terms have been added to the energy equation; (4) the user may now directly specify the amount of memory that is allocated for FLUINT matrix inversions (5) a new option has been added to the USRFIL routine intended to help users open and access unformatted (binary) user files, (6) a new utility routine ("CHOKER") has been added to simplify modeling of choked flow in paths, (7) new routines ("COMPLQ" and "WAVLIM") have been added to facilitate modeling of water hammer and acoustic waves, and (8) various fixes have been made in the tie (heat transfer) options, especially for tied junctions.

Changes to This Manual

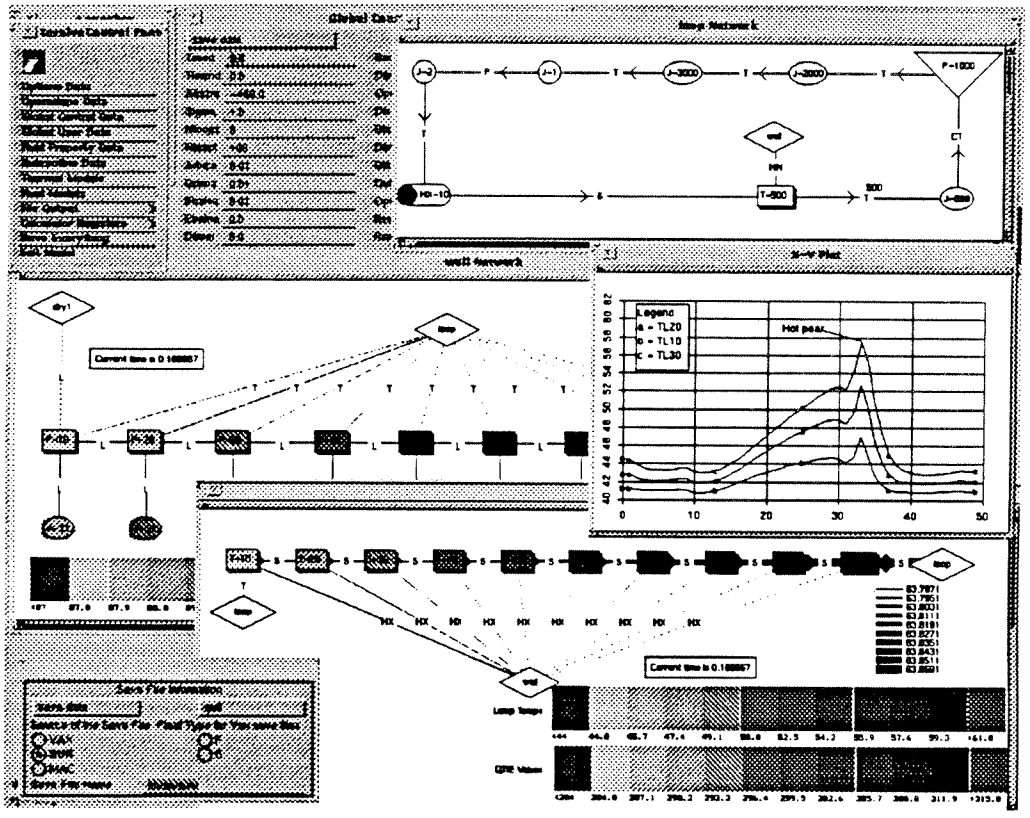
Users should review portions of the manual relating to the above improvements. As an aid to previous users, the following changes have been made to this manual:

- Updated sections .... 1.3, 3.13.6.3, 3.11.2 4.6, 4.7.3.2, 4.7.9.5, 4.7.14, 4.7.18.2, 4.7.19, 5.10, 5.11.4, 6.4.15, 6.9.1, 6.9.5, 6.9.6.5–6, App. E App. F (Sample Problems A, B, E, G)

  New sections 6.9.6.7

SINAPS: GRAPHICAL PRE- AND POST-PROCESSING

SINAPS (SINDA Application Programming System) is a companion program to SINDA/FLUINT that enables users to graphically sketch their models using a mouse- and menu-driven user interface. Forms and editing windows exist to satisfy other nongraphical input requirements. SINAPS then produces complete SINDA/FLUINT input files, and imports binary output files that were perhaps produced on other machines. This enables *graphical display of predictions on the same schematic used to create inputs*. In addition to pop-up X-Y, polar, and bar plots, features such as "color by flowrate," "thicken by conductance," and "shade by temperature" are supported. A sample SINAPS screen image is provided below:



SINAPS is a complete model management tool that eliminates the need to write ASCII inputs and interpret results via ASCII files. In fact, it contains many powerful features that are unavailable in the basic SINDA/FLUINT system, such as algebraic inputs, shared submodels, and custom components. To assist current SINDA/FLUINT users, it accepts existing ASCII input files, working interactively with the user to produce a graphical depiction of that model.

SINAPS is transportable. It was developed simultaneously on a Macintosh II and a SUN SPARCstation, has been rehosted on 386/486 PCs, and can be rehosted on most other Unix workstations. Perhaps more importantly, a SINDA/FLUINT model (and its graphical depiction such as that shown above) can be easily moved from one type of machine to another. Combining this feature with the fact that SINAPS and SINDA/FLUINT need not reside on the same machine gives the analyst tremendous flexibility.

SINAPS is documented in a separate User's Manual.

TABLE OF CONTENTS

PART A: Instruction Manual

Appendices

List of Tables

List of Figures

1. INTRODUCTION

1.1 SINDA: An Overview

SINDA is a software system suited for solving lumped parameter representations of physical problems governed by diffusion-type equations. The system was originally designed as a general thermal analyzer that utilizes resistor-capacitor (R-C) network representations of thermal systems; although, with due attention to units and thermally oriented peculiarities*, SINDA will accept R-C networks representing other types of systems.

SINDA was originally conceived and written under the name CINDA (Chrysler Integrated Numerical Differencing Analyzer) in the 1960's. Around 1970, the NASA JSC version of the program became SINDA and development began under a series of technology contracts. During the decade of the 1970's and into the early 1980's, SINDA has been improved by a series of enhancements that include the fluid flow network capability, improved network solution algorithms and the addition of a number of input options.

SINDA/FLUINT represents a more extensive change to the program than it has undergone in any one of the previous development changes. The only code recognizable from the previous SINDA is in the utility subroutine library such as the interpolation routines, and parts of the solution routines FORWRD, FWDBCK, and STDSTL. These routines are significantly changed where the "pseudo-compute sequence" is involved. The preprocessor is almost entirely new.

Users of old SINDA will, however, recognize commonality between SINDA/FLUINT and its predecessor, and should have no trouble using the program to accomplish anything that could previously be done. The basic hallmarks of SINDA's design have been retained. These are dual execution, using a preprocessor that a) accepts the user's definition of the R-C or flow network parameters and, b) converts user supplied "Fortran-like" language into Fortran. This Fortran is compiled and sent to the system loader. The loader collects the necessary parts of the precompiled Fortran subroutines that make up the processor library. Because many of the subroutines are invoked through user-supplied calls, each processor load module is essentially custom-designed by the user for the problem at hand. When the collect/load task is finished the resulting load module is executed. This is the so-called "processor execution."

A pressure/flow analysis of a system containing an arbitrary piping network can be performed simultaneously with the thermal analysis during transient or steady state solutions. This permits the mutual influences of thermal and fluid problems to be included in the analysis. This fluid network analysis capability is called FLUINT.

Within this manual, SINDA/FLUINT will always be regarded as a thermal and/or fluid network analyzer. The knowledgeable user will, however, recognize the program's ability to solve electrical networks or any other mathematical model that can be represented by a lumped-parameter network.

---

* For example, heat flow by radiation is unique to thermal systems in that it is a function of temperature to the fourth power. By contrast, no general flow phenomenon in fluid systems is a function of pressure to the fourth power.

1.2 SINDA/FLUINT Upgrades

The differences between old SINDA and SINDA/FLUINT are the result of the following list of imposed* top-level requirements:

- o Compatibility with previous SINDA inputs
- o Program transportability between different make computers
- o Compatibility with vector processing
- o Reduced run time
- o Network parameter identification by *submodel*
- o Convenient means of combining data from a number of sources into one input file
- o Convenient units control for both input and output
- o Addition or deletion of submodels during processor execution
- o Separate, addressable pseudo-compute sequences for each submodel
- o Division of time dependent updates and temperature dependent updates into separate VARIABLES blocks
- o Separate VARIABLES blocks for each submodel
- o Separate iteration counts for each submodel
- o Separate convergence criteria for each submodel
- o Separate output calls for each submodel
- o A SUBROUTINE DATA block for extended user logic
- o Convenient duplication of previously input blocks of data
- o Time-variable conductors
- o A graphics-oriented set of post-processor utility programs
- o Improved user-friendliness

It can be seen from this list that the envisioned program changes were not trivial nor necessarily easy to achieve. For instance, the submodel concept leads to an expansion in input complexity that works against the very important requirement for improved user-friendliness. Other contradictory requirements are expanded problem size and program transportability, which disallows the use of the tightly packed pseudo-compute sequence data storage that has always been a SINDA design feature. Some of the required increments in preprocessor capability work against preprocessor run-time reduction.

In general, the above list of requirements was met. Much larger memory sizes are available on mid-80's vintage computers relative to those of 1970, which allows larger models even with the unpacked pseudo-compute sequence. The unpacked pseudo-compute sequence enhances run time improvement in the processor. Other processor run time improvements were achieved by improving the structure of the iterating sections of the program solution code (important to vector processing) and by simply eliminating the solution routines that have proved to be inferior through past experience. Automatic time-step selection, now a default feature of the implicit transient solution routine, offers considerable potential for reduced run time. Preprocessor run time improvement was achieved by replacing the primitive data search/sort algorithms with much faster code, resulting in a faster preprocessor even though identification of each parameter by submodel adds to the preprocessor's task.

---

* NASA-JSC Request for Proposal 9-BC73-2-3-75P, *Development of an Advanced SINDA Thermal Analyzer Computer Program*, 19 May 1983.

In the opinion of the SINDA/FLUINT designers, a significant improvement in user-friendliness has been achieved. This is necessarily a subjective judgment, but elimination of some distinctly user-unfriendly aspects of SINDA could not help but improve matters. The more significant changes are:

- Elimination of column dependence in input.
- Elimination of superfluous input, such as END-cards.
- Fortran-syntax arithmetic allowed in data-value entries.
- Elimination of ambiguous syntax in the user logic inputs and improved syntax checks. The resulting language is more Fortran-related and thus more understandable to the new and infrequent user.
- The user is not required to know JCL (Job Control Language). Permanent file manipulation may be done within the program using program inputs.
- Program restart, initial parameters and final parameters features are all done outside the preprocessor, using subroutine calls that supply more flexibility along with straight-forward procedures.
- Logical program defaults should eliminate nearly all run failures due to insufficient program control inputs.

Experienced SINDA users will know that all this does not mean that input problems will now disappear. Any program that accepts input in a language akin to Fortran will never be easy to run except for elementary problems. The input language allows errors that can only be detected by the Fortran compiler or by the user when nonsense results are obtained. The improvements contained in SINDA/FLUINT will, however, eliminate much of the irritation caused by misplaced commas, missing END statements, data field violations and faulty translations to Fortran.

1.3 Introduction to the R-C Network

To introduce the user to the nature of the R-C network input required by SINDA, as well as to illustrate the basic flexibility of the program, a short sample problem, from engineering statement to completed input file, will be presented in the following paragraphs. Note that fluid networks (introduced in Section 4.7) are fundamentally different from the thermal networks that will be introduced here.

Consider a bar of metal as shown in Figure 1–1. It is 0.1 inch thick, 1.0 inch wide, and 3.0 inches long, and is fully insulated except for one end. A 10 watt heater is embedded at the insulated end of the bar, and the uninsulated end radiates to deep space. The bar is initially at a temperature of 70.0°F, and after turning on the heat, it is desired to know the time, to the nearest minute, when the center of the bar reaches a temperature of 200.0°F, and the temperature distribution in the bar at this time.

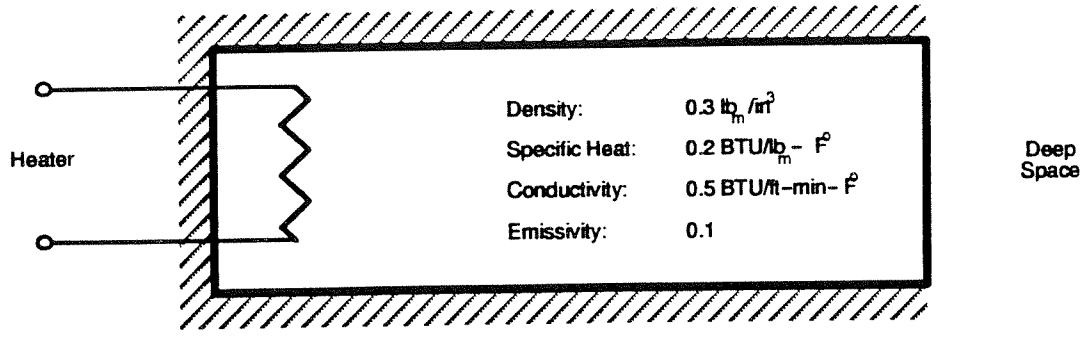


Figure 1-1 Bar of Metal for Sample Problem

Figure 1-2 shows the lumped parameter/R-C network which represents the stated problem. The mass of the bar has been divided into three* portions (shown schematically as capacitors) which are called *nodes*, and each node has been assigned an arbitrary identification number. The heat conduction paths, (shown schematically as resistors) which are called *conductors*, have also been assigned arbitrary identification numbers. A heat *source*, Q, which represents the heater in the insulated end of the bar, is shown entering node 10. Deep space is represented by a constant temperature node at -460°F (shown schematically as a ground).

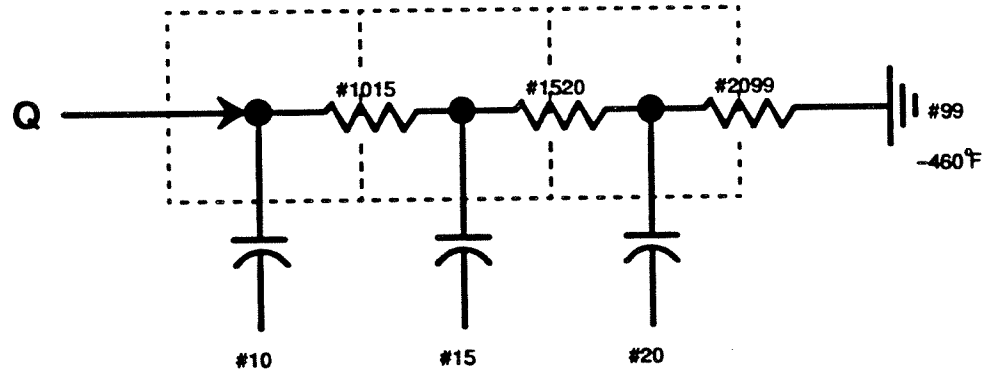


Figure 1-2 Sample Thermal Network

* An odd number of portions was chosen such that one would represent the mass at the center of the bar, which is of particular interest to this problem. Three chunks, rather than five or seven, etc., were selected on the basis of engineering judgment.

The thermal energy storage capacity (or *capacitance*) of each bar node is equal to the product of its density, volume, and specific heat (in this case, 0.006 BTU/°F). The three bar nodes may be defined by specifying their identification number, initial temperature, and capacitance as follows:

```
10, 70.0, 0.006                    record 1
15, 70.0, 0.006                    record 2
20, 70.0, 0.006                    record 3
```

The node representing deep space may be defined as follows:

```
-99, -460.0, 0.0                   record 4
```

The *conductance* of a conductor which represents a heat flow path through a material is equal to the product of the material's thermal conductivity and the cross-sectional area of the flow path, divided by the length of the path. The conductance of bar conductors 1015 and 1520 is, therefore, 0.00417 BTU/min-°F. The conductance of a conductor which represents heat flow by radiation is equal to the product of the Stefan-Boltzmann constant and the emissivity, area, and view factor from the surface. In this case, conductor 2099 has a conductance of 1.98×10^{-15} BTU/min-R^4. In SINDA, the three conductors may be defined by specifying their number, the numbers of their adjoining nodes, and their conductances as follows:

```
1015, 10, 15, 0.00417              record 5
1520, 15, 20, 0.00417              record 6
-2099, 20, 99, 1.98E-15            record 7
```

Since the bar will undoubtedly reach the temperature of interest in less than 1000 minutes and the output results need be accurate only to the nearest minute, the SINDA end time and output time control variables may be set, appropriately, as follows:

```
TIMEND = 1000.0, OUTPUT = 1.0      record 8
```

To impress the heat source on node 10, the following record is prepared:

```
10, 10.0*3.413/60.0                record 9
```

Since a transient analysis of the bar is desired, a routine from the SINDA library which performs transient analysis by the Forward-Backward (Crank-Nicholson) finite differencing technique will be called into action, with the following record:

```
CALL FWDBCK                        record 10
```

Finally, to suppress output until the center of the bar reaches 200°F, at which time it is desirable to print the temperatures of the nodes and to end the problem, the following logic statements are required:

```
IF (T15.GE.200.0) THEN                          record 11
  CALL TPRINT('SUB1') $ PRINT TEMPS             record 12
  TIMEND = TIMEN $ END PROBLEM NOW              record 13
ENDIF                                           record 14
```

Table 1–1 shows how the 14 records prepared specifically for the sample problem have been inserted in a "real" input file which contains the additional records necessary to separate, for example, the node data from the conductor data, etc. The list of records in Table 1–1 constitutes the complete SINDA/FLUINT input required to solve the stated problem.

The foregoing example has illustrated the complete process of transforming an engineering problem into a SINDA/FLUINT input deck. Only the most elementary features were employed so that the new user would not be overwhelmed with detail. At the same time, however, a small amount of program logic was included to expose the new user to the versatility and flexibility that are possible within the SINDA system.

Table 1–1 Complete Input File for Sample Problem

```
HEADER OPTIONS DATA
TITLE HEATED BAR SAMPLE PROBLEM
      OUTPUT = BAR.OUT
      MODEL = TEST
HEADER NODE DATA, SUB1
     10, 70.0, 0.006                $    record 1
     15, 70.0, 0.006                $    record 2
     20, 70.0, 0.006                $    record 3
     -99, -460., 0.0                $    record 4
HEADER CONDUCTOR DATA, SUB1
     1015, 10, 15, 0.00417          $    record 5
     1520, 15, 20, 0.00417          $    record 6
    -2099, 20, 99, 1.98E-15         $    record 7
HEADER CONTROL DATA, GLOBAL
     TIMEND = 1000.0, OUTPUT = 1.0 $    record 8
HEADER SOURCE DATA, SUB1
     10, 10.0*3.413/60.0            $    record 9
HEADER OPERATION DATA
BUILD TEST, SUB1
      CALL FWDBCK                   $    record 10
HEADER OUTPUT CALLS, SUB1
      IF(T15.GE.200.0) THEN         $    record 11
        CALL TPRINT('SUB1')         $    record 12
        TIMEND = TIMEN              $    record 13
      ENDIF                         $    record 14
END OF DATA
```

1.4 System Structure

It should be evident that the SINDA/FLUINT system is a complex applications program. It has, in fact, all of the functions and capabilities of a special purpose computer language. Since most computers in current use in engineering environments already have operating systems built around a Fortran compiler, the SINDA/FLUINT system is designed to augment the existing Fortran system. Hence, the SINDA/FLUINT library serves as an extension to the existing Fortran library, and the program serves as a preprocessor to (i.e., it precedes) the existing Fortran compiler. This augmentation arrangement is illustrated in Figure 1–3.

As can be seen in Figure 1–3, SINDA/FLUINT consists of two modules: an executable preprocessor and a library of processor routines. When a run is made, the preprocessor sorts through the user's inputs and generates both data files and Fortran source code files. The Fortran files are compiled and linked with the processor library, forming a dynamically dimensioned and customized executable program for every analysis. When the newly formed program is run, it immediately reads the data files left by the processor and begins to execute the user's instructions.

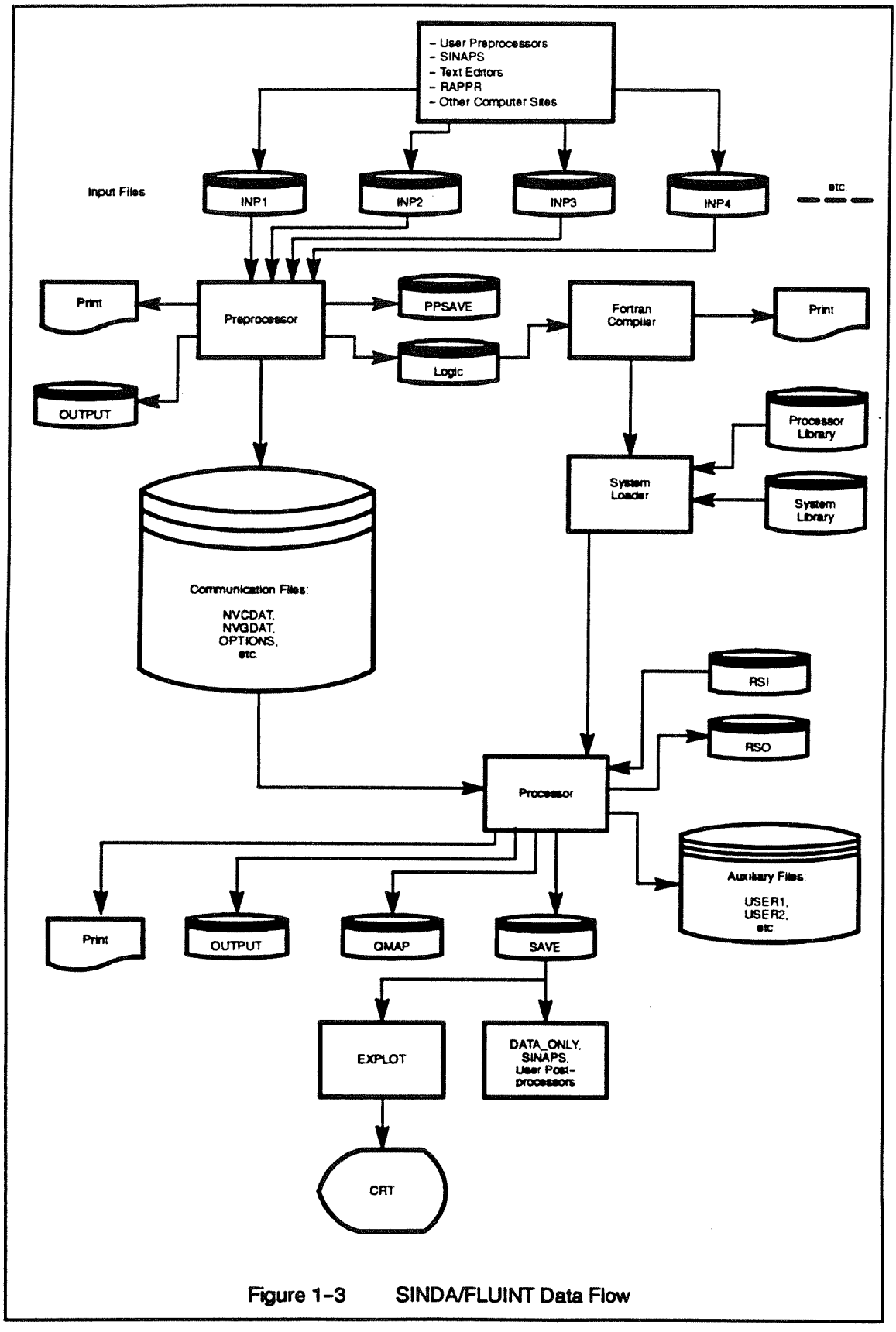


Figure 1-3 SINDA/FLUINT Data Flow

1.5 Model Size Limits

Release 2.6 of SINDA/FLUINT is dimensioned to handle a *total* of 20,000 nodes, 100,000 conductors and 10,000 arrays. For reference, the default size limits are listed below. For readers unfamiliar with SINDA/FLUINT, these terms will be defined in Sections 3 and 4.

THERMAL:	Submodels	100
	Nodes	20,000
	Conductors	100,000
FLUID:	Submodels	25
	Lumps	4,000
	Paths	4,000
	Paths per nonplenum lump	25
	Ties	6,000
	Macros	800
	User fluids	10
	Total length of FPROP arrays	2000
BOTH:	User arrays plus TARRAYs	10,000
	Values per array	10,000
	CARRAYs	1,000
	Named constants	1,000
	Numbered constants	1,000

These dimensions may be easily adjusted upwards or downwards to suit the needs of the user's application. To alter these limits, change the value of the desired parameter in the PARAMETER.INC "include file," and recompile all of the code. (A fundamental limit of 999 thermal and 999 fluid submodels exists.)

1.6 Learning SINDA/FLUINT

When using the full capability of SINDA/FLUINT, the engineer will be required to exert a programming effort of sorts, to a major extent in an Fortran-like SINDA language, and to some extent in Fortran. This dependence on programming, together with the wide variety of options and features offered by the system, suggests an appropriate word of caution. SINDA/FLUINT is a comprehensive system which cannot be mastered overnight. The prospective user should not assume that a cursory review of the Instruction Manual will lead to immediate success, nor should he assume that this manual represents a "cookbook" which will eventually yield to a plodding and rigid adherence to each and every rule. In presenting instructions on the use of a computer program, it is not possible to completely avoid some "cookbook-like" sections; however, every effort has been made to explain the "why" and "how" behind each rule, option, and feature, with the intent of encouraging the reader to think about and understand SINDA/FLUINT in depth. Without exception, every effort to *understand* SINDA/FLUINT will be rewarded by increased ability to *use* it.

2. BACKGROUND INFORMATION

2.1 Prerequisite Knowledge

In addition to a knowledge of heat transfer and fluid flow, the potential SINDA user will require some familiarity with the operation and programming of digital computers. Specifically, the user is assumed in this manual to have a working knowledge of the following topics:

- Computers
- Core Memory
- Permanent Files
- Data Storage (Tapes, Disks, Drums)
- Terminals
- Input Files
- Output Files
- Program Files
- Fortran Language
- Fortran Compiler
- Fortran Subroutines
- Text Editor

These topics will be referred to but not specifically explained herein. Users whose background includes an introductory course in digital computers and Fortran programming should be familiar with this list. Other users are referred to textbooks* on these topics.

2.2 Input Records

At this point, the old SINDA User's Manual included a section on punched cards and the convention used in the remainder of the manual to indicate the card column dependence of the various input data formats. This is not done here because punched cards have disappeared in favor of permanent files as the standard means of generating and storing input data. These files are generated at a computer terminal, using a text editor that is part of the computer's collection of "built-in" software (its operating system). In this manual, what was a punched card will now be referred to as an input data record or *record*. It is like a punched card in that it consists of about 80 characters, including blanks.

There is minimal discussion of how the following text will indicate card columns for the reason that, in SINDA/FLUINT, column dependence can be expressed by a single rule: column 1 is reserved for specific key characters and columns 2 through 80 normally constitute a free field.

---

* See for instance, J.W. Perry Cole, *ANSI Fortran IV with Fortran 77 Extensions*, Second Edition Wm. C. Brown Co., Dubuque, Iowa, 1983.

This rule will appear throughout the sections on input definition. The only exceptions to this rule are that columns 1 and 7 have the same significance as in any Fortran code, and a few special characters require two columns to be unique.

Except in logic blocks, where the Fortran 6th column rule applies, lines which are too long may be continued with a back slash (\). This \ must be the last character of the line to be continued. Thus, the following represents *one* input record:

first input ..... input \
input ..... input \
input ..... last input

Back slashes are not required in flow data and fluid property blocks, where a subblock input structure is used and inputs are expected to continue over many lines.

2.2.1 Format Definition Rules

This section describes the conventions that will be used when input data are indicated.

First, all input examples will be distinguished from normal text by using a larger typewriter font such as "FONT" and "`font`." Furthermore, capital and lower case type will assume a special significance. When an input record is defined in capital letters, the characters shown must appear in the record exactly as shown. For example, a record might be shown as follows:

```
NAME, RANK, SERIAL NUMBER, Q
```

The actual input record, correctly prepared, would then appear as follows:

```
NAME, RANK, SERIAL NUMBER, Q
```

On the other hand, lower case type is used to indicate a variable input: the user must supply something which corresponds to the item shown in lower case type. For example, an input may be specified as follows:

```
name, rank, serial number, q
```

where: `q` = length of service, (1 = 1 to 3 years,
2 = 4 to 6 years,
3 = greater than 6 years)

The record prepared according to this format might then appear as follows:

```
WILLIAM E. PISKE, CAPTAIN, A03096690, 1
```

Note that name, rank and serial number were considered self explanatory whereas q was not.

The two above examples of input records appeared in the text as non-indented lines. The significance of this is that such a record must begin in column 1, but no other column dependence need be observed. A familiar example of such a record is a comment "card" in a Fortran program. Such comments are also allowed in SINDA/FLUINT.

If the record appears indented, then it must begin to the right of column 1:

```
    NAME, RANK, SERIAL NUMBER, Q
```

When it is appropriate to identify a particular card format by placing a descriptive comment on the same printed line, this comment will always appear to the right of a dollar sign, $, as illustrated below:

```
ML, name, address      $(mailing list card format)
```

2.3 File and Tape Conventions

Since SINDA/FLUINT is implemented on a variety of computers, it is necessary to refer to data storage media by some nomenclature which will be independent of the particular system configuration. Fortran "logical unit numbers" are often used for this purpose, but were rejected for use in this manual because certain installations impose restrictions on the type of physical storage devices which may be assigned to a given unit. Instead, each serial access storage device referenced by SINDA/FLUINT is given a proper name such as "PURPOSE." Hence, for example, the OUTPUT file contains the printout resulting from processing the user's data, and the RSI file contains input for a restart run. Any file may, in fact, be a disk file, a drum file, or a magnetic tape, at the option of the user.

2.4 Terms and Data Conventions

2.4.1 Common Terms and Data Types

The word *subroutine* is generally used interchangeably with the word *routine*. *Integer* and *fixed point* mean the same thing, as do *real* and *floating point*. *Character* applies to ASCII character strings. The term *data value*, or *literal*, will be taken to mean one element of the set of all integers, floating point numbers, and multi-character strings.

Integers will be shown in print as a sequence of digits preceded, optionally, by a plus or minus sign. Floating point numbers will appear in print as a sequence of digits with a leading, trailing, or embedded decimal point, prefixed, optionally, by a plus or minus sign, and suffixed, optionally, by an exponent (to the base 10) denoted as the letter E followed by an integer. Character strings will be delimited in print by apostrophes ('), also called *single quotes*. These are necessary because blanks are valid characters and have a specific binary code (i.e., they do not appear on the printed page, but they do appear explicitly in the computer). Examples follow.

INTEGERS:

```
1                       +12345
-1                      -12345
12345                   15
```

FLOATING POINT NUMBERS:

```
1.0                     1.5E+6
 10.                    +1.5D6
.10                     -.18E-12
+.15                    -20.E-4
15.                     1467.812
```

CHARACTER STRINGS:

```
'1'                     '-1'
'ABC'                   '1.0'
'DOGS AND CATS'         '(+-+$)'
'    '                  '12.6'
```

In contrast to *data values*, another entity, called an *identifier* or *variable*, will be used (in a programming sense). For example, consider the following statement:

```
PI = 3.14
```

In this case, 3.14 is a floating point data value, and PI is an identifier. Note that PI is different from 'PI' which is a character string.

2.4.2 Computer Data Representation

The user must recognize that the three types of data—integer, floating point, and character—are quite different from the standpoint of the computer. To make this point clear, consider the following addition problem:

```
  100
   +100.0
'+100.0'
```

The answer, in human terms, is three hundred. However, a computer could not perform such a simple problem directly because its internal binary number system maintains three different modes of data representation. It has been the intent of this section to impress upon the reader that some of the same pitfalls awaiting the inexperienced Fortran programmer also await the SINDA/FLUINT user.

3. THE INPUT FILE

3.1 Introduction to the Input File

The SINDA/FLUINT input file is the means by which the user specifies the structure of his model and the method for solving it. The input file is subdivided into *blocks,* of which there are two types. *Data blocks* are used to pass numerical values to the program, and the input formats for such blocks are specific to the type of block being used. *Logic blocks* are used to pass executable instructions to the program in the form of a Fortran-like language. In fact, logic blocks are translated into real Fortran and then compiled and executed. With one exception noted below, blocks may occur in any order within the input file. The start of a new block is designated by the word "HEADER" starting in column 1, and the words after the HEADER command describe the type of block about to be input.

To specify the structure of the thermal mathematical model, three data blocks called NODE DATA, SOURCE DATA and CONDUCTOR DATA are provided. When fluid flow is involved, FLOW DATA blocks can be added (in which case the thermal model is optional).

Two fundamental kinds of program control are provided. One type of control uses program variables to control the operation of a pre-programmed SINDA/FLUINT module such as the preprocessor or a processor solution, math utility, or output routine. This parameter-style control is the domain of the OPTIONS DATA block, which is used primarily for preprocessor control, and the CONTROL DATA blocks, which are used primarily for processor control.

The second and most generalized type of program control is done through the logic blocks: OPERATIONS DATA, VARIABLES 0, VARIABLES 1, VARIABLES 2, FLOGIC 0, FLOGIC 1, FLOGIC 2, OUTPUT CALLS, and SUBROUTINE DATA. These blocks are all written in a SINDA-specialized variant of the Fortran language and provide the user with the capability to initialize parameters, specify his choice of solution method and then intervene by changing parameters at will during the solution process. For historical reasons, OPERATIONS DATA and SUBROUTINE DATA are actually misnomers; the user should remember that these are logic blocks, not data blocks.

In addition to model descriptions and program control, another type of input block is provided for the user to enter the data needed to solve the problem. These data consist of single values and arrays of values which can be integers, floating point numbers, or character strings. The blocks used for this data are USER DATA, ARRAY DATA, TARRAY DATA, CARRAY DATA, and FPROP DATA. USER DATA are single floating point or integer numbers declared by the user for later use in logic blocks. ARRAY and TARRAY DATA are arrays of floating point or integer numbers. CARRAY DATA are used to store strings of characters. FPROP DATA blocks are used to input alternate fluid property data for fluid submodels.

TARRAY DATA blocks are intended to contain time-dependent arrays which are considered too large to be stored in core memory in their entirety. As the time-dependent solution proceeds, only the two dependent-variable values closest to the problem time are in memory (one behind, one ahead). These two values provide the minimum information necessary for linear interpola-

tion and eliminate the need for huge blocks of central memory. This data block replaces the FLUXRD feature used in previous versions of SINDA. Note that the use of the TARRAY DATA block is not mandatory nor even recommended for time-dependent functions. Such arrays also may be entered in the ARRAY DATA block if they must be available in core memory. The ARRAY DATA block is also used for bivariate arrays, which may involve both time and temperature dependence.

The CARRAY DATA block is reserved for the user to enter character strings of up to 128 characters in length. This data is useful for output headings, plot titles, file names, and the like. Unlike previous versions of SINDA, there is no machine-dependent size limit on these character strings. The preprocessor automatically reserves the necessary space in core for these strings.

3.2 Input File Structure

The SINDA/FLUINT input file consists of any number of input blocks of the types introduced in the preceding section. They are separated by *header records*, similar to the BCD 3XXX---card familiar to SINDA users. The SINDA/FLUINT header records may contain arguments, so they are really macroinstructions to the preprocessor rather than simply dividers.

There are two blocks which must appear once, and only once, in an input file. These are OPTIONS DATA and OPERATIONS DATA. All the other blocks are optional. **The input file must begin with the OPTIONS DATA block.**

A SINDA/FLUINT model may consist of up to 100 thermal submodels and 25 fluid submodels. (These upper limits may be changed as noted in Section 1.5.) This means that the input file must contain one OPTIONS DATA block and one OPERATIONS DATA block, and may contain one SUBROUTINE DATA block and up to 100 each NODE DATA, SOURCE DATA, CONDUCTOR DATA, VARIABLES 0, VARIABLES 1 and VARIABLES 2 blocks; up to 25 each FLOW DATA, FLOGIC 0, FLOGIC 1, FLOGIC 2, and FPROP DATA blocks; up to 125 each CONTROL DATA, USER DATA, ARRAY DATA, TARRAY DATA, and OUTPUT CALLS blocks. There is no ambiguity for multiple blocks of any one type because each data value is linked with the submodel name that appears on the header record.

Submodel names may be any *unique* character alphanumeric string of up to 8 characters. In this case, "unique" precludes the use of reserved SINDA or FLUINT words (such as "T" and "ABSZRO" and "GLOBAL"), and no two submodels may share the same name, even if one is a thermal submodel and the other is a fluid submodel. This restriction also applies to any USER DATA or translated Fortran variable used in logic blocks (see Section 3.12.2).

3.3 Macroinstructions

Macroinstruction records all begin with a key character in Column 1. These records are few in number and powerful in function and should be well understood before attempting to generate SINDA/FLUINT input. Most of them have not been part of previous SINDA input, so experienced SINDA users should take note.

The following is a list of the allowed macroinstructions records. The character in parentheses must be in column 1.

(H)EADER	Records
(I)NCLUDE	Records
(C)OMMENT	Records
(R)EM	Records
(P)START	Records
(P)STOP	Records
(F)AC	Records
(B)UILD	Records
(B)UILDF	Records
(D)EBON	Records
(D)EBOFF	Records
(D)EFMOD	Records
(F)START	Records
(F)STOP	Records

The next eight sections present the format and function of these records.

3.3.1 Header Records

These records separate the input file into blocks, define submodel names and may contain multiplying factor data values where appropriate. Header record arguments apply to all the following input records until another header record is encountered or the input file ends. The majority of header records have the form:

```
HEADER type DATA {,smn} [options]
```

where `smn` is the submodel name (which may or may not be required for that type of block) and `type` is one of the following:

```
OPTIONS
CONTROL
USER
OPERATIONS
ARRAY
TARRAY
CARRAY
NODE
SOURCE
CONDUCTOR
SUBROUTINE
FLOW
FPROP
```

Square brackets [ ] indicate optional, defaulted arguments; braces { } indicate options that may or may not be required, depending on other inputs. The submodel name is *cannot* be used with OPTIONS DATA, OPERATIONS DATA, or SUBROUTINE DATA because these blocks only appear once. Nor can it be used with FPROP DATA which may appear several times but is not directly associated with any submodel. For other data types the submodel name *is* required.

Logic blocks, which take the format of Fortran-style inputs, omit the word "DATA" and always require a submodel designation (with the exceptions of the misnomers OPERATIONS DATA and SUBROUTINE DATA):

```
HEADER type,smn
```

where `type` is:

```
OUTPUT CALLS
VARIABLES 0
VARIABLES 1
VARIABLES 2
FLOGIC 0
FLOGIC 1
FLOGIC 2
```

A variant of the header record is used to fetch additional input records from permanent files on the computer's disk storage and insert them into the input stream. *Any data can be included anywhere by using the command:*

```
INCLUDE pfn,[ln1][:ln2]
```

where:

`pfn`* ..... valid permanent file name
`ln1` and `ln2` ..... starting and ending line numbers from pfn

This is particularly useful for array, source, and conductor data which are supplied by another program, and for FPROP data which may be reused in any model. To include a whole header section, the header record may be appended to the include command:

```
INCLUDE pfn,[ln1][:ln2] HEADER type {DATA}
    {,smn} [options]
```

One subtle rule of operation for the INCLUDE directive is that, upon returning from an INCLUDE operation, the program assumes it is still inside the same HEADER block in which the INCLUDE was encountered. This rule, consistent with subroutine-style operation, may cause problems when HEADER blocks are placed inside the INCLUDE file and/or a header record does not immediately follow an INCLUDE directive. The reader is referred to Section 4.4 for more details.

---

* The `pfn` argument in the INCLUDE record is the only place such files must be named. The SINDA '85 user is not required to attach such files to his run using the system's job control language.

3.3.2 Comments

Records beginning with a C or R in column 1 allow the user to arbitrarily insert comment data into the input file, as in Fortran code. Additionally, a dollar sign, $, signals the end of data on any line, allowing comments to be inserted afterward. However, the dollar sign should not be used on INCLUDE lines or OPTION DATA file name designation lines, otherwise it will be added into the file name. Also, the dollar sign should not be used on untranslated logic lines unless the compiler also recognizes this symbol (as does the DEC Fortran compiler).

3.3.3 PSTART and PSTOP Records

These records stop and start the preprocessor printback and save of the collected and sorted input file. The printout and save begins by default until a PSTOP is encountered, and a P- instruction stays in effect until another P- instruction is encountered. The various input printout and save options are explained in the OPTIONS DATA, Section 3.4. These instructions have no arguments.

3.3.4 FAC Record

One of the options in the NODE and CONDUCTOR DATA header record is `[,fac]`, a multiplier for capacitance and conductance data. FAC records are used to change the multiplying factor that is applied to all subsequent nodal capacitance or conductor data that is input using either the 3-blanks or the GEN option. The format is:

```
FAC dv
```

where `dv` is a floating point number.

The factor is set to 1.0 by default upon entering the NODE and CONDUCTOR DATA blocks, unless the `fac` option is used on the header record. The FAC record may not be used in any other data input block.

3.3.5 BUILD and BUILDF Instructions

These records define the currently active SINDA/FLUINT model in terms of a list of submodel names. They appear *only* in OPERATIONS DATA. *No submodel will be analyzed or even preprocessed unless it has been built; at least one BUILD or BUILDF instruction must appear in OPERATIONS DATA before calls to any solution or output routine.* The format is:

```
BUILD mn, sn1, sn2, . . . sni, . . . snn
```

where:

`mn` ..... arbitrary configuration name
`sni` ..... *thermal* submodel names, 1 to 8 characters.

The BUILD commands are translated into Fortran code. If more than one 80 column record is required for the sequence, continuation characters are placed in column 6. The configuration name must be to the right of column 6. A HEADER NODE DATA record must be present for every submodel name on the BUILD list. The NODE DATA block may be empty in the case of submodels consisting of conductors only.

Any subsequent BUILD operations erase rather than append to the previous BUILD operation. However, any submodel not currently active will retain the status it had when last active. Thus, results are never lost (barring the use of parametric or restart options).

BUILDF instructions operate analogously for *fluid* submodel names. To build both types of submodels into one configuration, use the same configuration name on adjacent BUILD and BUILDF instructions, as depicted in the following example:

```
BUILD CONFIG, THERM1, THERM2
BUILDF CONFIG, FIRST, SECOND,
     + THIRD
```

Models cannot be explicitly unbuilt. Once built, they can only be unbuilt by another BUILD or BUILDF instruction that does not include that submodel. Thus, to unbuild either all thermal models or all fluid models, a fake (nearly empty) submodel must be input and built.

3.3.6 DEBON and DEBOFF Instructions

These commands start and stop the debug mode in the preprocessor. If the debug mode is on, the preprocessor is directed to search all logic statements without an F in column 1 for local variables. Local variables are variables *not* stored in program common (e.g., entered in a data block). Any local variables found will generate an error message. Examples of valid program variables are T, Q, G, and any CONTROL or USER DATA variable. The debug mode is a tool primarily used to spot variables that are not defined or are perhaps misspelled. The debug mode should be used to prevent insidious Fortran errors. Its use forces the user to declare valid variable names in USER DATA. Such declarations are mandatory in most programming languages.

A DEBON instruction is terminated when a header card is encountered or a DEBOFF record is encountered. These instructions have no arguments. Note also that the debug feature may be turned off globally in OPTIONS DATA by stating NODEBUG.

3.3.7 FSTART and FSTOP Instructions

These instructions direct the processor to copy *verbatim* all logic records included between FSTART and FSTOP without attempting any conversions or translations. This is equivalent to placing an F in column 1 for all statements between the FSTART and the FSTOP commands. This option is most useful in the SUBROUTINE DATA block to prevent any conversion from taking place when a Fortran routine is inserted.

Care should be exercised in the use of these instructions in logic blocks, however, since they may prevent desired conversions. Significant improvements in SINDA/FLUINT conversions over those of older SINDA versions, on the other hand, make it unlikely that the preprocessor will attempt to erroneously translate any valid Fortran expressions.

3.3.8 DEFMOD Instruction

This instruction sets or resets the current default submodel name. It can be used only in OPERATIONS and SUBROUTINE DATA where the default submodel name is 'GLOBAL'. The format is:

```
DEFMOD smn
```

The use of this option eliminates the need for the repetitive input of a model name. For example, "MODL.T100" could be input as "T100" after a DEFMOD MODL record.

The default submodel name may be either a thermal or fluid submodel, but only one submodel name may be defaulted at any time.

3.4 OPTIONS DATA

All SINDA/FLUINT input files must begin with a single OPTIONS DATA block. The only mandatory parts are the header record and the one-line problem title. All other variable inputs are optional and defaulted if appropriate. The variables defined in the OPTIONS DATA are flags that (1) control the preprocessor's output, (2) name permanent files for program use, and (3) perform miscellaneous functions like model name definition and debug/no debug control.

Figure 3–1 and Table 3–1 are a preprocessor execution flow chart and a reference summary of the OPTIONS DATA inputs. The remainder of the text in this section is for user guidance in preprocessor control. (Table 3–3 is an example of an OPTIONS DATA block.)

3.4.1 Permanent File Definition

Table 3–2 presents the SINDA/FLUINT program file definition and usage. Any of these files may be saved as a permanent file on any host system by entering the generic file mnemonic (e.g., QMAP, SAVE) along with an arbitrary name for the permanent file in OPTIONS DATA. If files are not specifically named, some machines (VAX and most Unix machines) will save output files anyway under system-selected names such as FORT.002. Users are cautioned that some platforms (e.g., HP/Unix) do not have default file names.

Except for the OUTPUT file, none of these inputs are defaulted, and no read/write action will occur unless the name appears in association with a permanent file name in the OPTIONS DATA block. The default for OUTPUT is to append the processor's printout to the preprocessor printout file, as defined in the runstream (job control).

The OPTIONS DATA block is the *only* place in the user's input stream where it is *necessary* to name a permanent file (e.g., RSI, SAVE) and define its use within SINDA/FLUINT. This is different from old SINDA where it was necessary for the user to be familiar with the computer's operating system commands. The INCLUDE directive, if used, will also contain permanent file names. The user should not use the dollar sign ($) style comments on the same line as OPTIONS DATA or INCLUDE file names. A similar restriction applies to file names input as CARRAY DATA for use in the USRFIL and CRASH utilities introduced below.

The user may optionally open two user files USER1 and USER2, and refer to those files as unit numbers NUSER1 and NUSER2, respectively. If more files are needed, they may be opened by calling the USRFIL routine (see Section 6).

Another routine opens a file is the optional CRASH routine, which saves recent snapshots of the model as frequently as desired without requiring excessive disk space (see Section 6).

3.4.2 Other Options

When the DEBUG flag is on (the default condition), the preprocessor examines each variable name found in the logic blocks to see if it matches one of the user accessible program variables, (e.g., T, Q, C, TL, PL, TIMEND, DRLXCA, etc.) or a user input array or variable name. If not, the preprocessor will raise an error flag and point out the offending variable name. See Section 3.12.3.4 for more information on this feature.

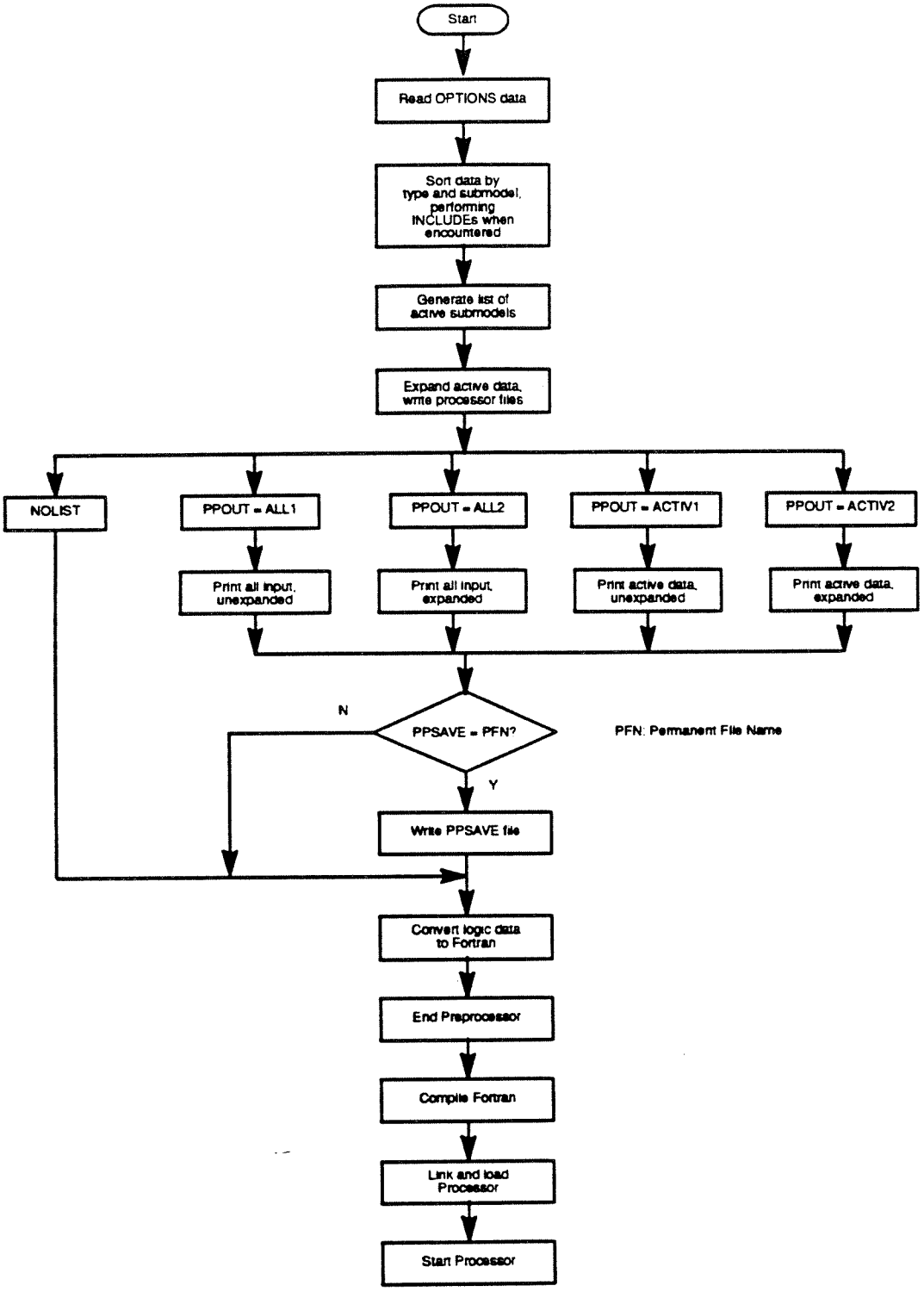


Figure 3–1 Preprocessor Flow Chart

Table 3-1 OPTIONS DATA Summary

Variable Name	Description and Options	Default
TITLE	Problem title– up to 72 alphanumeric characters	None – mandatory input
PPSAVE	Input save file – any permanent file name (pfn) up to 8 characters	None – optional input
RSI	Restart input file – any pfn	None – optional input
RSO	Restart output file – any pfn	None – optional input
OUTPUT	Processor print file – any pfn	Controlled by JCL (same as preprocessor print file) System print file name
SAVE	File for TSAVE & SAVE data – any pfn	None – optional input
QMAP	File for QMAP and FLOMAP data – any pfn	None – optional input
USER1 USER2	User auxiliary files – any pfn	None – optional input
MODEL	Model name, up to 8 characters	'ROOT'
PPOUT	Input data print/save control flag. ALL1, ALL2, ACTIV1, ACTIV2	ALL2 – Save all input, expanded
NOLIST	No printout flag	None – optional input
MLINE	Number of printed lines per page (integer)	54 – Univac, 58 – VAX
NODEBUG	User logic debug flag	Off – optional input
DIRECTORIES	Print flag for node, conductor, array, lump, path, tie, and user constant directories	Off – optional input

Table 3-2 Program File Definitions

File Name/No.	Function/Content	Writing Subroutine	Reading Subroutine	Type
INPUT/5	OPTIONS DATA, other data blocks at user's option	None	Preprocessor	Formatted
INCLUDE files	Any number of files containing data called in by INCLUDE cards	None	Preprocessor	Formatted
PPSAVE/39	The file containing entire model after INCLUDE operations	Preprocessor	Preprocessor	Formatted
SAVE/21	File containing all program variables and variable name dictionaries. Used for restart with RESAVE, for boundary temperature driving with BNDDRV, and for post-processor programs such as EXPLOT and DATA_ONLY	SAVE TSAVE	BNDGET EXPLOT DATA_ONLY etc	Binary
RSO/25	Identical to SAVE file. Allows user to write two different sets of output from same run	RESAVE	RESTAR EXPLOT etc.	Binary
User Defined	A program file generated by BNDGET for use by BNDDRV. This file may be saved as a permanent file if desired	BNDGET	BNDDRV	Formatted
QMAP/18	File containing QMAP (thermal network values and connectivity) and/or FLOMAP (fluid network values and connectivity) output	QMAP FLOMAP	None	Formatted
USER1/22 USER2/23	Files provided for user output	User logic	User logic	Formatted
PRINT/6	The printed output file	Preprocessor and Processor	None	Formatted

The DIRECTORIES flag turns on printout of the node, conductor, lump, path, tie, array, and constants "directories" or mappings that show the correspondence between the user's identification numbers and the internal (relative) numbering scheme. The MLINE variable allows the user to define the number of lines per page in the printout. The MODEL name option is for the user's convenience.

The PPOUT variable defines the content of the preprocessor's printout of the input data. Five options are provided:

`NOLIST .....`	No printout
`PPOUT = ALL1 .....`	Lists input records verbatim from all input files, sorted by data type & submodel, GEN records not expanded.
`PPOUT = ALL2 .....`	Same as ALL1, with GEN records expanded. (Default condition)
`PPOUT = ACTIV1 .....`	Same as ALL1 except only the portions of the input that will be active during the run are printed.
`PPOUT = ACTIV2 .....`	Same as ACTIV1, with GEN records expanded.

If a PPSAVE permanent file is defined, a copy of the printout (as defined by PPOUT) will be written to PPSAVE and saved. This file is intended to be an orderly, sorted file for the user to edit in the system text editor as the model is updated. There is a potential problem here because the file planned for editing may become very large, especially if individual GEN expansion records are present. The text editor may be very slow or simply will not take the entire file. For this reason, the INCLUDE macroinstruction, in conjunction with the PSTART & PSTOP macroinstructions, should be used to keep large input files separated into conveniently sized parts. Computer generated radiation conductor files should probably always remain as INCLUDE files. See Section 3.3 for information on these features. Finally, note that FLUINT macros, the equivalent of thermal GEN options, are *not* expanded into the PPSAVE file because the relationship between generated elements would be lost if they were input separately.

The user also has control over the presentation of output material via the MLINE and TITLE keywords. These values be altered during processor execution by calling the PAGHED routine (see Section 6).

3.4.3 OPTIONS Data Formats

Table 3–3 is an example of an OPTIONS DATA block that illustrates all options.

Table 3–3 OPTIONS DATA Block Example

```
HEADER OPTIONS DATA
TITLE    OPTIONS DATA EXAMPLE
         MODEL      =    TEST
         PPSAVE     =    PFN1
         PPOUT      =    ALL1
         RSI        =    PFN2
         RSO        =    PFN3
         OUTPUT     =    PFN4
         SAVE       =    PFN5
         QMAP       =    PFN6
         PLOT       =    PFN7
         USER1      =    PFN8
         USER2      =    PFN9
         MLINE      =    45
         NODEBUG
         DIRECTORIES
         NOLIST
```

3.5 CONTROL DATA

CONTROL DATA blocks are used to exert user control over the problem solution by initializing a set of control "constants" that the program stores in memory under reserved variable names. The quotation marks are to indicate that these values are not constant at all, and may be initialized or changed at will in the user's logic blocks—the historical name "constants" is a misnomer. Examples of these variables are TIMEND and NLOOPS, the problem stop time and maximum steady state iteration count allowed, respectively.

SINDA/FLUINT models that consist of more than one submodel require both single-valued control constants and arrays of control constants. Single-valued constants apply to all submodels and will be familiar to SINDA users. Multiple value control constants are arrays of values N long where N is the number of currently active* submodels. If the same value applies to all submodels of the same type (i.e., thermal or fluid), the user can define one value to apply to all submodels. This global value can then be overridden for each submodel if desired, as shown below.

3.5.1 Defining CONTROL Data Values

Global control constants are entered under a single HEADER CONTROL DATA, GLOBAL record. *Note that this precludes naming a submodel 'GLOBAL'.* Any control variable name in Table 3–4 may be initialized in this block by a single statement. For instance, in a model consisting of five thermal submodels, the single statement OUTPUT = 0.1 would set five output time interval array elements to 0.1. Later, under a HEADER CONTROL DATA, FRED record, the statement OUTPUT = 0.01 would set the output time interval to 0.01 for submodel FRED. *Note that the GLOBAL block must precede the other control data blocks* because the last input overrides all previous inputs.

3.5.2 CONTROL Data Names and Functions

Table 3–4 is a list of all the SINDA/FLUINT program control "constants," a brief description of their function, whether they may be multi-valued or not, and their defaults, if any. It is only necessary for the user to define all non-defaulted values if both steady-state and transient solutions are required. The use of some of the control constants can be quite involved and subtle, and usually dependent on their use with particular solution routines. The user is referred to Section 4.6 for more information on this subject.

3.5.3 CONTROL Data Formats

When a HEADER CONTROL DATA record is encountered, the preprocessor expects a series of one or more "NAME = value," fragments in each data field (columns 2 thru 80) on the following records until another header card is encountered. Such fragments must not be split between records (lines). Table 3–5 is an example of the CONTROL DATA blocks for a two submodel, transient solution model.

---

* An active submodel is one whose name appears on a BUILD or BUILDF macroinstruction.

Table 3-4 Control Constant Definitions

NAME	DEFINITION	OPTIONS	DEFAULT
Single-Valued Constants: Input under submodel 'GLOBAL' only			
TIMEO	Initial time	Real	0.0
TIMEND	Stop time	Real > TIMEO	NONE
NLOOPS	Maximum iteration count allowed (steady state)	Positive integer	NONE
DTIMES	Pseudo time step (steady state)	Real	0.0
ABSZRO	Value of absolute zero in user's temperature units	Real	-459.67 (ENG) -273.15 (SI)
PATMOS	Value of absolute zero in user's pressure units (FLUINT)	Real, double precision	0.0D0
SIGMA	Stefan-Boltzmann constant	Positive real	1.0
UID	Unit system identifier	SI or ENG (note: no quotes)	ENG (English units)
ACCELX/Y/Z	Acceleration vector components (FLUINT)	Real	0.0
Multiple-Valued Constants: Input one data value per thermal submodel			
ARLXCA	Maximum temperature change allowed for convergence of arithmetic nodes (degrees)	Positive real	0.01
NLOOPT	Maximum iteration count allowed (transient)	Positive integer	100
DRLXCA	Maximum temperature change allowed for convergence of diffusion nodes (degrees)	Positive real	0.01
EBALSA/ EBALNA	Steady state submodel/node energy balance required for convergence - EBALSA is fraction of total model energy flow, EBALNA is absolute	Positive real, 0.0 to 1.0 (EBALSA) 0.0 = off (EBALNA)	0.01/0.0
ATMPCA	Maximum temperature change allowed per time step for arithmetic nodes (degrees)	Positive real	1.0E30
DTMPCA	Maximum temperature change allowed per time step for diffusion nodes (degrees)	Positive real	1.0E30
DTIMEL	Minimum allowable time step (trasient).	Positive real	0.0
DTIMEH	Maximum allowable time step (transient)	Positive real	1.0E30
DTIMEI	Specified time step (transient)	Positive real	0.0 (automatic time step calc)
ITEROT	Iteration interval for OUTPUT CALLS (steady state)	Positive integer	0 (print upon convergence)
OUTPUT	Time interval for OUTPUT CALLS (transient)	Positive real	NONE
ITHOLD	Number of iterations a submodel can remain in semi-converged boundary state (steady state)	Positive integer	10
EXTLIM	Maximum allowable extrapolation temperature change	Positive real	50.0
ITERXT	Number of iterations bewteen extrapolation	Integer, > 2	3
CSGFAC	Time step factor for explicit transients	Real, > 1.0	1.0
BACKUP	Flag to repeat time step (transient)	Real, 0.0 or 1.0	0.0 (no repeat)
OPEITR	Flag to print after each time step (transient)	Integer, 0 or 1	0 (no print)
Multiple-Valued Constants: Input one data value per fluid submodel			
DTSIZF/ DTTUBF	Transient time step size factor; fractional change allowed in key variables (transient). DTTUBF is the fractional change allowed in tube flowrates.	Positive real 0.0 to 1.0, exclusive	0.1
DTMAXF	Maximum time step (transient)	Positive real	1.0E30
DTMINF	Minimum average time step (transient)	Real	0.0
OUTPTF	Fluid submodel version of OUTPUT	Positive real	NONE
OPITRF	Fluid submodel version of OPEITR	Integer, 0 or 1	0 (no print)
RSSIZF/ RSTUBF	Same as DTSIZF for STDSTL pseudo transient relaxations. RSTUBF parallels DTTUBF.	Positive real 0.0 to 1.0, exclusive	0.5
RSMAXF	Same as DTMAXF for STDSTL; maximum artificial time step	Positive real	1 hour
RERRF	Maximum relative change allowed per iteration in PL, TL, XL, and FR (steady state)	Positive real	0.01
REBALF	Fluid submodel version of EBALSA	Real	0.01
ITROTF	Fluid submodel version of ITROUT	Positive integer	0
ITHLDF	Fluid submodel version of ITHOLD	Positive integer	10

Table 3-5 CONTROL DATA Block Example

```
HEADER CONTROL DATA, GLOBAL
    NLOOPS    =      50
    DRLXCA    =       0.01,  ARLXCA = 0.02   $Note: closing comma understood
    ABSZRO    =  - 459.69
    EXTLIM    =      10.0
    TIMEO     =       0.2
    TIMEND    =       5.0
    OUTPUT    =       0.2
HEADER CONTROL DATA, FRIC1
    EXTLIM    =     10.0,    DRLXCA = 0.005
HEADER CONTROL DATA, FRIC2
    OUTPUT    =       1.0
```

When multiple CONTROL DATA blocks are required, they need not appear together. The preprocessor will gather them together before interpreting them. *The GLOBAL block must, however, appear only once and precede all the other CONTROL DATA blocks.*

3.6 USER DATA

User constants differ from control constants. (Again, "constants" is a historical misnomer.) User constants are all optional and are never referenced by any preprogrammed part of the processor. Rather, they are referenced and used only in other portions of the model. Beginning with SINDA/FLUINT, user constants are separated from control constants, primarily for better input data organization.

The user may or may not identify particular constants with a particular submodel, at his option. The "GLOBAL" model name is provided for *named constants* (e.g., 'FRED' or 'FISH') that are not referenced using a "real" submodel name as an identifying qualifier. Submodel-specific *(numbered)* constants have predefined names with user-input identifiers. *As with CONTROL DATA, the global block must precede the submodel blocks even though the two types of constants appear unrelated.*

3.6.1 Global (Named) Constants

Global user constants are single-value program variables (stored in common blocks) that are labeled by any unique symbolic variable name that is not a program reserved name (see Section 4.6). The variable names are supplied along with each constant's initial value in the USER DATA blocks under submodel GLOBAL. The following input rules apply:

1) This block must precede the other USER DATA blocks.

2) Only symbolic name identifiers of up to 8 characters with an alphabetic lead character are allowed, e.g.: ITEST, XMAX1. Numeric identifiers are not allowed.

3) Only one GLOBAL block is allowed. Each global constant name must be unique.

4) Integer identifiers must be entered with integer data values in order to initialize the data value as an integer. The same with real identifiers/data values. *The implicit Fortran identifier rule applies.* That is, identifiers beginning with I, J, K, L, M or N will be treated as integers or fixed point. All others are real or floating point.

5) Character variables are not allowed. CARRAY DATA blocks, described later, are provided for this.

6) The names ATEST, BTEST, ... ZTEST are available by default, although they may also be explicitly set. ITEST through NTEST are integers, the remainder are real.

The only input format allowed for global user constants is "VN,dv" pairs, separated by commas or equal signs, that may appear anywhere between columns 2 and 80. Pairs must be complete on a line. An example of a global USER DATA block is as follows:

```
HEADER USER DATA, GLOBAL
     NTEST = 10
     MTEST = 20
     OMEGA = 0.707/2. + 1.0
     PI = 3.1416
```

Floating point data values may be defined in terms of any Fortran arithmetic statement, providing that only +, -, *, or / are involved. No functional relationships or parentheses will be accepted. *Note that Fortran-style arithmetic operations proceed left to right with no precedence given to * or / operations.* Operation on two integers produces an integer result; otherwise all numbers are treated as floating point.

3.6.2 Referencing Global User Constants

Global user constants may be referenced by name or redefined in any M-type or F-type statement in the logic block. Given a global USER DATA block containing the variables PTEST, QTEST, RTEST, ITEST and JTEST (which are all available by default) the following are all examples of references that may appear in any of the OPERATIONS DATA, VARIABLES n, FLOGIC n, OUTPUT or SUBROUTINE DATA blocks:

```
RTEST = ZPOD.G(1201)*QTEST
CALL DA11MC (1.512,ZPOD.TIMEM, ZPOD.A(112),
+ PTEST, ZPOD.Q(101))
DO 111 ITEST = 1, JTEST
```

References to global user constants are restricted to the logic blocks. When constant references are required in NODE, CONDUCTOR or SOURCE DATA blocks, they must be numbered constants (see below) associated with a specific submodel.

3.6.3 Numbered (Submodel-Specific) User Constants

Numbered user constants are defined using:

```
HEADER USER DATA, smn
```

where `smn` is a "real" (as opposed to "global") submodel name.

Numbered constants are given an arbitrary integer identifier by the user (e.g., "10", or "80201"). The full identification of a numbered constant includes the submodel name. Thus, the integer identifiers need only be unique within a single submodel block. The same numbered constant entered under a separate and distinct submodel name can still be uniquely referenced. As each constant is read in by the preprocessor, it is assigned an internal sequence number. This sequence encompasses the entire set of numbered constants identified in all submodels. (This same treatment applies to other network elements to be defined later: nodes, conductors, lumps, paths, ties, and arrays.)

User constant references, outside the USER DATA blocks, are of the forms:

```
[smn.]XKn or [smn.]XK(n)          $ real constant
[smn.]Kn  or [smn.]K(n)           $ integer constant
```

where:

`smn` ..... submodel name

`n` ..... actual (user's) identification number

References of this type are allowed in any of the logic blocks (OPERATIONS DATA, VARIABLES n, FLOGIC n, OUTPUT CALLS and SUBROUTINE DATA) as well as within the NODE, CONDUCTOR and SOURCE DATA blocks as allowed by certain options described in later sections. Note that user constants cannot be referenced in FLOW DATA or FPROP blocks.

User constants may be initialized with integer or floating point (real) data values. Character data is input separately in CARRAY DATA blocks, described in later sections. Previous versions of SINDA allowed constants to be identified with a fixed set of variable names (e.g., ITEST, RTEST, etc). This feature has been expanded to allow user-supplied variable names, but such constants must appear under the submodel name "GLOBAL." See Sections 3.6.1 and 3.6.2 above. The largest user constant identification number allowed is 999999, and a maximum of 1000* user constants (total from all active submodels) is allowed.

Most advanced users will eventually need to know how numbered user constants are stored internally in the processor, although such knowledge is not strictly necessary for the casual user. Each submodel's set of numbered user constants are stored contiguously in input order, and all sets are stored contiguously in one array—in fact, integer and real constants share the same array (named K or XK) via a Fortran EQUIVALENCE statement. The preprocessor stores the starting sequence numbers associated with each submodel and the size of each set. These data allow the processor to unambiguously access any numbered user constant.

It is sometimes useful to know that the constants occupy sequential core locations in the same order as they are input. For instance, a user might access or update a series of constants using a Fortran DO loop if the appropriate sequence numbers are known. This information is available for user constants and for other sequentially stored data types in the form of user *directories*, which may be printed using an OPTIONS DATA flag. If the appropriate directory is not available, the user may employ one of the utility subroutines that produce the internal sequence number, given the user's actual number. A separate routine is available for user constants, arrays, nodes, conductors, lumps, ties, and paths. (Reference Section 6.6: NODTRN, CONTRN, INTLMP, etc.) *Use of these "dynamic translation" routines is recommended not only to avoid memorizing storage sequences, but also to avoid insidious errors caused by changes to or maintenance of the model, which will also change the storage directories.*

3.6.4 Input Formats: Numbered User Constants

The standard input format for numbered constants is:

```
      id = dv
or    id, dv
```

where:

`id` ..... integer identification number

`dv` ..... integer or floating point data value

* On some compilers, this number may be reduced to about 200. A preprocessor caution is issued in cases where a compiler limitation may be reached. Section 1.5 describes how to change limits.

Multiple id-dv pairs, separated by commas, may appear within the column 2-80 data field over several lines, but each pair must be contained on a single line.

The GEN input option allows the user to generate and initialize a group of numbered constants having the same initial value or sequentially incremented values. Two GEN formats are allowed:

```
GEN id#, #k, ik, dv
GEN id#, #k, ik, dvi, idv
```

where:

id#	user identification number of the first constant in the series (integer)
#k	number of constants to be generated (integer)
ik	identification number increment desired (integer)
dv	single data value (integer or real)
dvi	data value to be assigned to initial constant id# (integer or real)
idv	increment to be added to data value to obtain the next consecutive data value in the set (integer or real)

Several GEN groups may appear on the same record, separated by commas. Groups must be complete within columns 2 through 80 (i.e., not separated over two lines).

3.6.5 Input Examples: Numbered User Constants

The following examples illustrate numbered constant data input:

```
HEADER USER DATA, POD1
     101=3.485*144.+14.7, 201 = 864 $ Standard Inputs
C Generates 10 zero value constants
     GEN 901, 10, 1, 0.
C Generates 3 constants with values 100., 150., 200.
     GEN 1001, 3, 1, 100., 50.
```

Data values may be defined in terms of any Fortran arithmetic statement provided that only +, -, * or / are involved. No functional relationships or parentheses are allowed. *Note that Fortran-style arithmetic operations proceed left to right with no precedence given to * or / operations.* Operation on two integers produces an integer result; otherwise all numbers are treated as floating point.

3.7 ARRAY DATA

ARRAY DATA is analogous in function to numbered USER DATA. The difference is that the data is in multi-element sets stored in sequential core locations rather as single words.

Two new blocks have been added to the single ARRAY DATA block allowed in previous versions of SINDA. The CARRAY DATA blocks provide a way to input character data. The TARRAY DATA blocks are restricted to singlet arrays that define the time-dependent relationships of model parameters such as heat sources and conductance. All of the TARRAY data is never in core at once, only the data values necessary to support interpolation for the current point in simulated time. The standard ARRAY DATA block is still available for temperature-dependent data, bi- and trivariate arrays and any other miscellaneous types the user may require. Array data storage and access rules have not changed drastically with SINDA/FLUINT. An upgrade is that there is now no need to supply the word END to terminate an array.

Because of the use of arithmetic operations (+,-,*,/) in ARRAY DATA blocks, *the user must use commas between data values. Spaces are unacceptable as delimiters, but are not flagged as errors.* Otherwise, "10.0 -5.0" will be interpreted as a single data value of 5.0 (=10.0-5.0) rather than two values of 10.0 and -5.0. This represents a deviation from old SINDA formats, in which arithmetic operations were not possible.

All data meant to be treated as real numbers must have an explicitly input decimal point, or else they will be treated (and stored) as integers even in the middle of an array of real values. Both this mistake and forgetting the comma between values can lead to insidious problems, and neither can be trapped as input errors without severely restricting other program features and options.

3.7.1 Standard Array Data

Standard array data is entered as numbered arrays using:

```
HEADER ARRAY DATA, smn
```

where `smn` is a submodel name. The standard format for inputting arrays is:

$$\texttt{num} = dv_1,\ dv_2,\ - - -\ dv_n$$

where:

`num` ..... Array ID number
dv_n ..... n^{th} data value in array

Note that there is no END terminator required to signal the end of the array. The "=" character following the array number signals the end of any previous array. Thus the "=" cannot be replaced by a comma as in previous versions of SINDA. Arrays from old SINDA models, containing the END terminators, may be input if the user desires. However, *the user may not mix old and new input forms in the same block.* Array data records must all be within the columns 2 through 80 data field and may continue for as many lines as necessary. Lines of array data are automatically terminated with a comma if one is not provided by the user.

Array identification numbers up to 999999 are accepted. The maximum number of standard arrays, (including character arrays, Section 3.7.2) is 10000. Data values may be integer or floating point and mixes are acceptable. Up to 10000 arrays may be entered in the TARRAY blocks. Character data must be input in the CARRAY blocks, described later. However, the total of TARRAYs plus standard arrays must not exceed 10000, and no single array may have more than 10000 elements. Up to 10000 character arrays may be input. (Section 1.5 describes how to change such limits.)

The user may reserve words in memory within an array using the SPACE input option. Such words will be initialized to 0.0. Whenever the fragment: "SPACE, m" is encountered *within* the data value field of a numbered array, m data values of zero will be inserted.

3.7.2 Array Data Structures

The previous section presented the general rules for entering numbered array data. Specifically structured arrays are required to support a number of program options. These include specifically structured singlet arrays, doublet arrays, bivariate and trivariate arrays. The specific formats for entering these specialized arrays are presented in this section.

Singlet Arrays — Singlet arrays contain a sequence of data values of the same type representing various values of the same parameter. The order of the values within the array is peculiar to its particular usage. For example, the singlet array required for polynomial evaluation of a variable capacitance node (see Section 3.8.1) contains floating point coefficients in the order of increasing powers of the independent variable, as shown below:

$$P_0, P_1, P_2, \ldots, P_n$$

Doublet Arrays — This type of array, most often required for interpolation options and subroutines, contains a sequence of ordered pairs of values usually representing points on a plane curve. The general order for the data values in a doublet array is as follows:

$$X_1, Y_1, \; X_2, Y_2, \; \ldots, X_i, Y_i, \; \ldots, \; X_n, Y_n$$

where:

X ..... independent variable
X_i ..... (i = 1,2,...,n) is strictly increasing with i.
Y ..... dependent variable

All interpolation options and subroutines require X and Y to be floating point numbers. A more memory-efficient method of entering X-Y curve data is available when there are a number of curves defined by dependent variable data values that each correspond to a single set of independent variable data values. This is the situation when a radiation analysis program is used to generate flux vs. time data for a large number of nodes that comprise an orbiting spacecraft. The first data value in each Q-array will be for the first time point in orbit, the second for the

next, and so on. In this situation, entering the data as doublet arrays would mean that each time-value would be unnecessarily repeated in each array. When this situation exists, enter the independent variable data once as a singlet array. The dependent variable data is also entered as singlet arrays and interpolation is done with a subroutine that utilizes two singlet array references as arguments (e.g., DA11MC).

Bivariate Arrays — This type of array is used to represent a function of two independent variables: Z = f(X,Y). Data values for bivariate arrays are input in the following order:

$$
\begin{array}{l}
n,\ X_1,\ X_2,\ \ldots,X_n \\
Y_1, Z_{11}, Z_{12}, \ldots, Z_{1n} \\
Y_2, Z_{21}, Z_{22}, \ldots, Z_{2n} \\
\cdot \qquad\qquad\qquad\qquad \cdot \\
\cdot \qquad\qquad\qquad\qquad \cdot \\
\cdot \qquad\qquad\qquad\qquad \cdot \\
Y_m, Z_{m1}, Z_{m2}, \ldots, Z_{mn}
\end{array}
$$

where:

n	Number of X values (integer)
m	Number of Y values (this value is not input explicitly)
Z_{ji}	$f(X_i, Y_j)$
X, Y, Z	floating point values
X_i	(i = 1,2,...,n) is strictly increasing in i.
Y_j	(j = 1,2,...,m) is strictly increasing in j.

The value of m is not input explicitly because the value of n (input as the first data value) and the value of the integer count (generated by the preprocessor) are sufficient to define the location of any element in the array. The following ARRAY DATA cards define bivariate array number 6 which contains values for the function Z = X + Y, taken from the graph in Figure 3–2:

```
6 = 3,     1.0,3.0,5.0
           2.0,3.0,5.0,7.0
           4.0,5.0,7.0,9.0
```

Although the sample cards above are more readable, the array could just as well have been input as follows:

```
6 = 3,1.0,3.0,5.0,2.0,3.0,5.0,7.0,4.0,5.0,7.0,9.0
```

Note that there is no need for a line-ending comma.

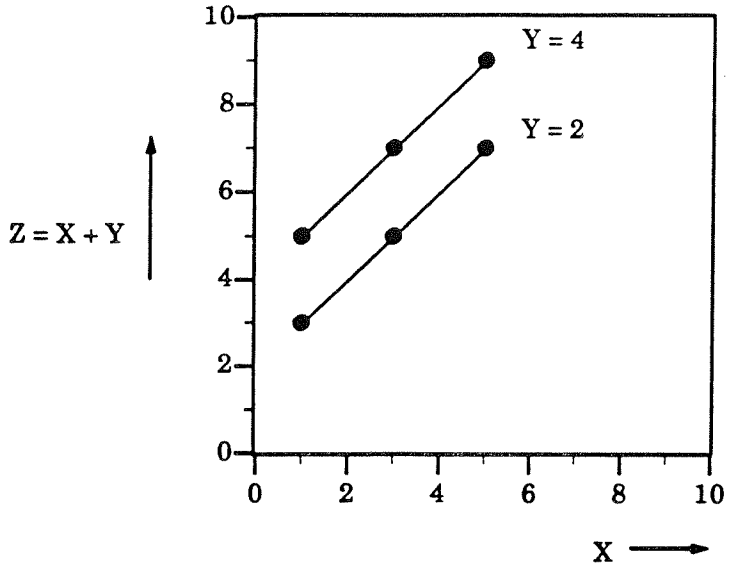


Figure 3-2 Sample Bivariate Function

Trivariate Array — This type of array may be thought of as two or more bivariate arrays, where each bivariate array is associated with a third independent variable. Trivariate arrays are used to represent functions of the form F = f(X,Y,Z,) for the purpose of evaluating such functions by interpolation. The data values in a trivariate array are input in the following order:

$NX_1, NY_1, Z_1, X_1, X_2, \ldots, X_n$
$Y_1, F_{11}, F_{12}, \ldots, F_{1n}$
$Y_2, F_{21}, F_{22}, \ldots, F_{2n}$ $ bivariate "sheet" for Z_1
$Y_m, F_{m1}, F_{m2}, \ldots, F_{mn}$
$NX_2, NY_2, Z_2, X_1, X_2, \ldots, X_j$
$Y1, F_{11}, F_{12}, \ldots, F_{1j}$
$Y_2, F_{21}, F_{22}, \ldots, F_{2j}$
$Y_k, F_{k1}, F_{k2}, \ldots, F_{kj}$
$NX_3, NY_3, Z_3, \ldots, \ldots, \ldots, \ldots$

A trivariate array may contain as many bivariate "sheets" as desired. The number of X and Y values in each sheet must be specified as integers NX and NY, respectively. NX and NY need not be the same for all sheets: All values X_i, Y_i, Z_i and F_{ij} must be floating point numbers. The following ARRAY DATA cards define a trivariate array which represents the function F = X + Y + Z over limited ranges of the independent variables:

```
10 = 3,2,1.0,1.0,2.0,3.0      $ (array number 10)
                 2.0,4.0,5.0,6.0
                 3.0,5.0,6.0,7.0
     2,3,2.0,2.0,3.0
                 2.0,6.0,7.0
                 3.0,7.0,8.0
                 4.0,8.0,9.0
     4,2,3.0,2.0,3.0,4.0,5.0
                 0.0,5.0,6.0,7.0,8.0
                 1.0,6.0,7.0,8.0,9.0
```

CARRAY DATA — Character data are entered using:

```
HEADER CARRAY DATA, smn
```

Character arrays are numbered "arrays," each containing a single data value. Such data values consist of up to 128 alphanumeric characters. The following is an example of CARRAY input:

```
HEADER CARRAY DATA, GEORGE
     120 = SAMPLE CHARACTER ARRAY
```

The data value consists of all characters following the equal sign (=) on that line. *Only one CARRAY value can be input on each line.* All character arrays are stored as 128 characters, blank filled on the right.

TARRAY DATA — (Note: TARRAY DATA is intended for use on nonvirtual machines with core memory limitations. Its use is otherwise neither necessary nor recommended.) The TARRAY DATA blocks are provided for large numbers of dependent variable vs. time arrays, where, for each array, each dependent variable data value corresponds to a member of a single set of time values. This leads to the following set of rules for TARRAY data:

1) Each array under a TARRAY header must be a singlet array of the same length.

2) One of the singlet arrays must be a time array, entered in monotonically increasing order of time.

3) The remainder of the arrays must be Q or G arrays with their data values corresponding exactly with the data values in the time array.

Standard array data is read by the preprocessor and written to the ARYDAT file in the format that is ultimately placed in the processor. TARRAY Data is written to the TRYDAT file in a rearranged format so that only appropriate portions of it are placed in memory at one time. When a TARRAY header is encountered, the preprocessor first needs the identification numbers that appear ahead of each array and writes them to TRYDAT as a string of integers in the order they are encountered. The first data value of each array is then read and written to TRYDAT the same way. This is repeated for the second through last data values.

When a solution routine does a linear interpolation to obtain a Q or G value at a given time point, all the array data values are superfluous except for the one corresponding to the time points just greater than and just less than the time for which a Q or G value is required. The TRYDAT data arrangement is specifically designed for the interpolation routine to quickly bring into core the two required data values.

This data arrangement makes it impossible for the user to access individual TARRAY data values as described in Section 3.7.3 below. If arrays are to be used for purposes other than linear interpolation, they must be entered in the standard ARRAY DATA blocks. The only interpolation routines that can use TARRAY data are DT11MC and D11MDT. Using any of the others will cause an error.

3.7.3 Referencing Array Data

All standard numerical array data, regardless of input structure, are stored a Fortran array named "A." A is equivalenced to an array named "NA" so that either integer or real data values may be accessed in core without data translation problems peculiar to Fortran. An array defined in an input record as:

$$n = v_1, v_2, \ldots v_m$$

resides in memory within the A array as

$$m, v_1, v_2 \ldots v_m$$

where m is the integer count (i.e., the total number of v values in that array).

Array references are required in the NODE, CONDUCTOR, SOURCE, and FLOW DATA blocks and in the logic blocks. Two types of references are required. The reference may be to the array as an entity or merely to a single element in the array. The two reference forms are

```
A(n) or An          $(Array as an entity)
```

and

```
A(n+e)              $(Accesses a data value)
```

where:

n ..... array identification number (integer)
e ..... numerical position of a specific element in the array (integer)

The A(n) form "points" to the array's integer count in the A array. This is sufficient data for a subroutine to use the array, if the arrays structure matches the subroutine's expectations. The A(n+e) form simply points to an element in the A array. Note that this procedure is really no different from accessing a user constant.

The following are examples of array data access forms that may appear in logic data:

```
C - - - Example 1:  Subroutine call in VARIABLES 0
      CALL DA11MC(PER,TIMEM,A(120),A(131),Q(980))
```

The subroutine is supplied with references to a time array, array 120, and a Q-source array, array 131.

```
C - - - Example 2: Referencing a floating point value
      TIMEP = TIMEP + POD.A(131+6)
```

Element no. 6 of array 131 of submodel POD is accessed.

```
C - - - Example 3: Referencing an integer data value
      ISTEP = M+POD.NA(200+2)
C - - - Example 4: Referencing an integer count
      IC = POD.NA(300)
```

The integer count of array 300 will appear in IC. Note: IC = POD.A(300) is an error.

3.7.4 Referencing Character Array Data

Each line of character array (CARRAY) data is stored as an entry into the UCA array. The reference form is:

```
UCA(n)  or UCAn
```

where n is the array identification number (integer)

Each UCA entry is a left justified 128 character string. This string can be used for any operations in which a character string is required.

3.8 NODE DATA

Nodes are fundamental SINDA network elements that describe how energy is to be stored or otherwise conserved. Four types of nodes may be defined in the NODE DATA block: *diffusion*, *arithmetic*, *boundary*, and *heater*.

Diffusion nodes have finite thermal mass or *capacitance*, and thus store and release energy. In the transient network solution routines, diffusion node temperatures are calculated by a finite difference representation of the partial differential heat transfer equation. Three locations in core memory are reserved for each diffusion node in an active submodel: one location to store the temperature of the node, one to store its capacitance, and one for the heat source* (if any) impressed on the node. Diffusion node data input options are provided to accommodate capacitance values which are not constant (i.e., vary with temperature, etc.)

Arithmetic nodes have zero capacitance. The temperatures of arithmetic nodes are calculated using a finite difference representation of Poisson's equation**. Since there is no capacitance value to store, only two core locations are reserved for each arithmetic node: one for the temperature, and one for the impressed heat source (if any).

Boundary nodes have infinite capacitance and may not receive an impressed heat source. A single core location is reserved to store the temperature of each boundary node. The temperatures of boundary nodes are not altered by the network solution routines, but may be modified as desired by the user.

Heater nodes are used for the purpose of establishing heater power requirements. They are the same as boundary nodes except that they report the amount of heat removed or added to maintain the given temperature. This energy is calculated using calls to HNQCAL.

3.8.1 Node Data Input

All NODE DATA records appear after:

```
HEADER NODE DATA, smn [,fac] [,tcon]
```

where:

`smn`	submodel name
`fac`	capacitance multiplying factor
`tcon`	temperature conversion flag
`[ ]`	indicates optional argument

* As with SINDA, these source locations are used by the solution routines as net heat rates. **The impressed heat rates must be reinitialized each time step if they are changed at all.**

** That is, arithmetic nodes are brought into a steady state heat balance according to the source applied and the heat exchange with neighboring nodes.

The submodel name is any identifying name, up to 8 characters. `tcon` is a code word used to convert the temperature on the NODE DATA records from one units base to another, if desired. Allowable `tcon` options are:

```
FTC ..... Fahrenheit to Centigrade
CTF ..... Centigrade to Fahrenheit
FTR ..... Fahrenheit to Rankine
RTF ..... Rankine to Fahrenheit
CTK ..... Centigrade to Kelvin
KTC ..... Kelvin to Centigrade
FTK ..... Fahrenheit to Kelvin
KTF ..... Kelvin to Fahrenheit
CTR ..... Centigrade to Rankine
RTC ..... Rankine to Centigrade
RTK ..... Rankine to Kelvin
KTR ..... Kelvin to Rankine
```

Please note that it is the user's responsibility to ensure that all submodels are consistent as to temperature and other dimensional parameters. Temperature units are controlled by `tcon` and control constant ABSZRO. (See Section 4.6: Advanced Control Constant Usage.)

There are several signals or codes which may be used on NODE DATA records. The codes provide options which enable the user to input a large variety of nodes in a simple, convenient manner. For all options, the following points apply:

1) Diffusion nodes must be given a positive node number and a positive capacitance.

2) Arithmetic nodes must be given a positive node number and a negative (*not zero*) capacitance. (The negative capacitance value acts as a signal that the node is of the arithmetic type.)

3) Boundary nodes must be given a negative node number and a non-negative capacitance (e.g., 0.0). The capacitance value is ignored but is present for the sake of consistency, and the negative sign on the node number serves as a flag to signify that the node is of the boundary type.

4) Heater nodes must be given a negative node number and a negative capacitance.

5) An initial temperature must be specified for all nodes, regardless of type.

6) Actual node numbers may be as large as six digits (e.g., 999999).

Table 3–6 summarizes the NODE DATA input options which are available to the user. Before proceeding to a detailed discussion of each option, two points should be clarified: (1) *impressed heat sources are not input with node data*; they are input in the SOURCE DATA block, in which case they are transferred to the source locations automatically when needed; and (2) nodal capacitances and heat sources are always accessible to the user in the logic blocks, and hence, a "constant capacitance value," from the standpoint of NODE DATA input, need not be held constant during the entire course of a solution.

Table 3-6 Summary of NODE Data Input Options

OPTION (CODE)	NODE TYPE D	A	B	H	DESCRIPTION
3 blanks	■	■	■	■	To input a single node where the capacitance is given as a single, constant value
CAL	■				To input a single node where the capacitance will be calculated by the preprocessor from four factors input by the user
GEN	■	■	■	■	To generate a group of nodes, each having the same initial temperature and the same capacitance
SIV SPV	■				To input a single node where the capacitance varies with temperature For SIV, the capacitance is found by interpolation using an array of temperature vs. capacitance. For SPV, the capacitance is found by computing an Nth order polynomial function of temperature
SIM SPM	■				To generate a group of nodes, each having the same initial temperature and the same temperature–varying capacitance. For SIM, C is found by interpolation using an array of T vs.C. For SPM, C is found by computing a polynomial in T
DIV DPV	■				To input a single node consisting of two materials which have different temperature–varying capacitances. For DIV, C1 and C2 are taken from arrays of T vs. C. For DPV, C1 and C2 are computed from polynomials in T
DIM DPM	■				To generate a group of nodes, each of which consists of the same two materials having temperature–varying capacitances. For DIM, C1 and C2 are taken from arrays of T vs. C. For DPM, C1 and C2 are computed from polynomials in T.
BIV	■				To input a single node where the capacitance is a function of time and temperature. The capacitance is found by interpolation using an array of time and temperature vs. capacitance

Standard Option ("3 blank" card code) — The simplest NODE DATA record utilizes the blank code. Records using this code must contain three data values for each node, as follows:

```
N#,Ti,C                    $(basic format)
5,70.0,0.75*2.0            $ example 1
6,80.0,-1.0                $ example 2
-9,90.0,0.0                $ example 3
-7,70.0,0.0,8,85.0,-1.0    $ example 4
```

where:

N# ..... Actual node number assigned by the user (integer data value)
Ti ..... Initial temperature of the node (floating point data value)
C ..... Capacitance of the node (floating point data value)

The order and type of the three data values must always be exactly as shown. The rules determining the type of input node are as follows: if N# and C are positive, then the triplet defines a diffusion node; if N# is positive and C is negative, an arithmetic node is defined; if N# is negative and C is zero, then a boundary node is defined. More than one node may be defined on a single card by placing more than one set of three data values on it.

Example 1 creates a diffusion node with an actual node number of 5, an initial temperature of 70.0°, and a capacitance of 1.5. Example 2 creates arithmetic node number 6 with an initial temperature of 80.0°. Example 3 creates boundary node number 9 *(not -9: the negative sign*

serves only as a flag!) with an initial temperature of 90.0°. Example 4 shows how the data values for two nodes may be placed on the same record. The two nodes created in example 4 are as follows: boundary node number 7 at a temperature of 70.0°, and arithmetic node number 8 at a temperature of 85.0°.

CAL Option — The SINDA feature that allows Fortran arithmetic (limited to +,-,*,/) in data fields makes the CAL input option obsolete. The CAL option is available but not documented for SINDA/FLUINT.

GEN Option — The GEN option is used to generate and input a group of nodes, each having the same initial temperature and the same capacitance. This option requires five data values for each group of similar nodes, as follows:

```
GEN N#,#N,IN,Ti,C          $ (basic format)
GEN -20,3,5,70.0,1.2       $ example 5
```

where:

N# ..... Actual node number of the first node to be generated (integer)
#N ..... Total number of nodes to be generated (integer)
IN ..... Increment to be added to the current node number to form the node number of the next node (non-zero integer)
Ti ..... Initial temperature of all nodes (floating point)
C ..... Capacitance of all nodes (floating point)

If C and N# are positive, a group of diffusion nodes will be generated. If C is negative a group of arithmetic nodes is produced. If N# is negative, then the nodes will all be of the boundary type.

It is important to remember that the negative sign on the N# serves only as a flag to identify a boundary node during input. For reference purposes, the negative sign is deleted. Hence, example 5 will generate three boundary nodes having a temperature of 70.0°, with the node numbers being 20, 25, and 30 (not -20, -15, and -10). The minus sign on the initial N# signifies that the entire group will be boundary nodes. A minus sign in the initial N# and the initial C signifies that the entire group will be heater nodes.

The eight-data-value GEN option, like the CAL option, has been made obsolete by the Fortran arithmetic feature of SINDA/FLUINT. It is available but not documented.

Automated Variable Capacitance Options — As stated earlier, the network solution routines call for the execution of the operations in the VARIABLES 1 block prior to performing each iteration on the heat transfer equations. VARIABLES 0 is called after each time step update in a transient solution. Using this feature, the user can set time or temperature variable capacitances by specifying VARIABLES 1 or VARIABLES 0 operations which would update values in the node capacitance table. The new capacitance could be calculated using the appropriate algorithm, or interpolation subroutine and independent variable. Though straightforward, this approach tends to be rather cumbersome. It is, however, more computer time efficient than the automated options described next.

The automated options cause temperature-dependent capacitances to be calculated wholly within the network solution routines, at the end of the VARIABLES 1 operations, and therefore relieve the user of the task of preparing and inputting capacitance computation algorithms. As specified by the user, one of two methods is used to update the current value of a temperature-dependent capacitance: (1) interpolation, and (2) polynomial evaluation. For interpolation, the user must input, in an ARRAY DATA block, a doublet array of (temperature, capacitance) ordered pairs representing points on the curve of T vs. C. The capacitance is calculated by performing linear interpolation on this array using the current temperature of the node as the independent variable. For polynomial evaluation, the user must input a singlet array of coefficients (P_0, P_1, P_2, ..., P_n) in an ARRAY DATA block. The capacitance is then calculated by evaluating the n^{th} order polynomial in T (i.e., P_0+P_1*T+P_2*T**2+...+P_n*T**n). These two methods enable the automated options to be applicable to a vast majority of the variable capacitance nodes encountered in modeling real systems.

To enable the same array of points or coefficients to be used for all nodes made of the same material, all automated options cause the computed capacitance to be multiplied by a factor before being inserted in the capacitance table. In this way, for example, the array can be used to compute ρ*C_p, and the multiplying factor can be used to specify the volume. In another application, the multiplying factor might be used to scale the capacitance to some set of consistent units.

In all cases, the multiplying factor may be input as a floating point data value, or as a user constant reference of the form XKm, Km, smn.XKm, or smn.Km, where m is the actual constant number. If the latter form is used, the referenced constant must be a floating point value.

The interpolation options (except BIV*) require a doublet array of temperature vs. capacitance to be input in an ARRAY DATA block. (Capacitance is used loosely in this context. The array could contain, for example, values of temperature vs. specific heat, as long as the associated multiplying factor had units of mass.) The array must contain an even number of floating point data values, as follows:

$$T_1, C_1,\ T_2, C_2,\ \ldots,\ T_i, C_i,\ \ldots,\ T_n, C_n$$

The temperature values, T_i, must be strictly increasing with i. If the node temperature at some time is less than T_1, then C_1 will be used in lieu of performing an extrapolation; if the temperature is greater than T_n, then C_n will be used.

The polynomial options require that an array of coefficients be input in an ARRAY DATA block. The array must consist solely of n+1 floating point values, where n is the order of the polynomial. The coefficients should be entered in the array in the order of increasing powers of temperature (e.g., the first data value is the constant term, P_0, and the $(n+1)^{th}$ data value is the coefficient, P_n, of T**n). For both classes of options, the arrays are referenced by using the form An or smn.An where n is the array number.

* The BIV option requires a bivariate array of time and temperature vs capacitance.

SIV and SPV Options — These options allow the user to define single nodes having temperature varying capacitances. The SIV option assumes that the referenced array is to be used for linear interpolation, whereas the SPV option assumes that the array contains polynomial coefficients. Both options require four data values for each node, as follows:

```
SIV N#,Ti,AP,F          $ (basic format)
SPV N#,Ti,AC,F          $ (basic format)
SPV 25,70.0,A8,0.0001   $ example 6
SIV 25,70.0,A9,0.0001   $ example 7
SPV 25,70.0,A8,K5       $ example 8
SIV 25,70.0,A9,XK5      $ example 9
```

where:

N# ..... Actual node number (integer)
Ti ..... Initial temperature of node (floating point)
AP,AC .... Reference to an array of T vs. C points, or polynomial coefficients, respectively
F ..... Multiplying factor input as a floating point data value, or a user constant reference of the form: XKm, Km, smn.XKm, or smn.Km, where m is the user constant number

The following example is presented to clarify the usage of these two options. Consider a temperature varying capacitance given by the equation:

$$C = (0.0001) \times (0.005 \times T^2 + 1.0)$$

This equation represents the curve shown in Figure 3–3. To use the SPV option to define node 25, which has a capacitance given by the equation above, the user would prepare a NODE DATA record as shown in example 6. The referenced array of polynomial coefficients, user array number 8, would be entered in an ARRAY DATA block, as follows:

```
8 = 1.0,0.0,0.005
```

To use the interpolation option, SIV, to define this node, the user would prepare a node data record as shown in example 7. The referenced array of T vs. C points, A9, represents a piecewise linear approximation of the curve in Figure 3–3, and would be entered in the appropriate ARRAY DATA block as follows:

```
9 = 0.0,1.0,10.0,1.5,20.0,3.0,40.0,9.0
```

Examples 6 and 7 show that the multiplying factor, 0.0001, was entered directly on the NODE DATA record as a floating point data value. This factor could also have been entered as a user constant in a USER DATA block, in which case, it would be referenced by its actual constant number on the NODE DATA records (see examples 8 and 9). The required USER DATA record would appear as follows:

```
5 = 0.0001
```

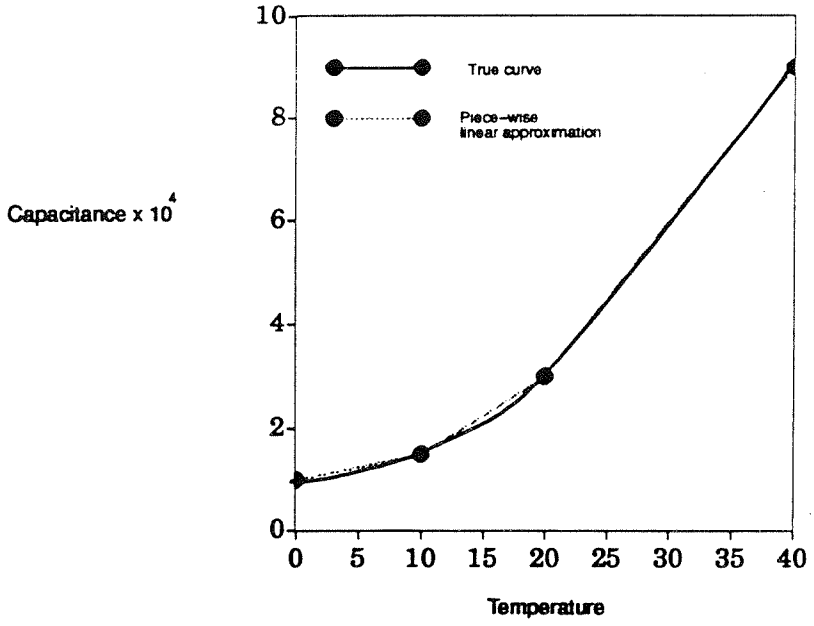


Figure 3-3 Curve of Temperature-Varying Capacitance

A common usage of these options is to input an array of $\rho * C_p$ vs. T (i.e., density times specific heat versus temperature) *once* for each material used in the model, and then to specify the volume of each node as the multiplying factor in each node declaration.

SIM and SPM Options — These two options combine the features of the SIV and SPV options with those of the GEN option. That is, they allow the user to generate and input a group of nodes all having the same initial temperature and the same temperature varying capacitance. The SIM option generates nodes whose capacitance will be evaluated by interpolation, and the SPM option generates nodes whose capacitance will be evaluated from a polynomial. Six data values are required to define each group of nodes, as follows:

```
SIM N#,#N,IN,Ti,AP,F          $ (basic format)
SPM N#,#N,IN,Ti,AC,F          $ (basic format)
SIM 25,3,5,70.0,A9,K5         $ example 10
```

where:

N#	Actual node number of the first node (integer)
#N	Total number of nodes to be generated (integer)
IN	Increment to be added to the current node number to form the number of the next node (non-zero integer)
Ti	Initial temperature of all nodes (floating point)
AP,AC	Reference to an array of points of T vs. C, or polynomial coefficients, respectively. An or smn.An form, where n is the array number)
F	Multiplying factor (floating point data value or user constant reference of the form: XKm, Km, smn.XKm, or smn.Km, where m is the constant number)

Example 10 defines three nodes, numbers 25, 30, and 35, whose capacitance will be computed by interpolating on array 9, with the results multiplied by constant 5.

DIV and DPV Options — These options allow the user to define a node which is made of two materials, each having different temperature varying capacitance properties. Both require an array and a multiplying factor for each of the two materials. For the DIV option, the result of interpolating on the first array, times the first multiplying factor, is added to the result of doing likewise with the second array and the second multiplying factor. The action of the DPV option is similar, except that the arrays are assumed to contain polynomial coefficients. The formats are shown below:

```
DIV N#,Ti,AP1,F1,AP2,F1          $(basic format)
DPV N#,Ti,AC1,F1,AC2,F2          $(basic format)
DIV 40,70.0,A8,K5,A6,2.0         $ example 11
```

where:

N#	Actual node number (integer)
Ti	Initial temperature (floating point)
AP1,AP2 ..	References to arrays of T vs. C points (An or smn.An form, where n is the array number)
AC1,AC2 ..	References to arrays of polynomial coefficients (An or smn.An form)
F1,F2	Multiplying factors (floating point data value, or user constant reference of the form XKm, Km, smn.XKm, or smn.Km, where m is the constant number)

Example 11 defines node 40, whose capacitance will be computed as the sum of interpolation on array 8 times constant 5, plus interpolation on array 6 times 2.0.

If one of the materials does not have a temperature varying capacitance, the user may substitute a floating point data value for one of the array references of the NODE DATA card. This substitute value will then be used in place of the result of interpolation (or polynomial evaluation) on the corresponding array. This feature makes the following alternate record formats possible:

```
DIV N#,Ti,SUB,F1,AP2,F2       $(alternate format)
DPV N#,Ti,SUB,F1,AC2,F2       $(alternate format)
DIV N#,Ti,AP1,F1,SUB,F2       $(alternate format)
DPV N#,Ti,AC1,F1,SUB,F        $(alternate format)
DIV 35,70.0,1.0,K5,A9,0.003   $ example 12
```

where SUB is a floating point data value.

Example 12 illustrates the case where one of the two materials has a constant capacitance. Since the capacitance of the first material will be calculated as 1.0*K5, it would be quite acceptable to alter the value of constant K5, at some point in an operations block, and thereby effect a change in the capacitance of this "constant capacitance" material.

DIM and DPM Options — These two options combine the features of the DIV and DPV options with those of the GEN options. They are used to generate and input a group of nodes, all of which have the same initial temperature and all of which are of the same two materials having temperature varying capacitances. The DIM option causes the capacitance for the two materials to be evaluated by interpolating on the referenced arrays, and the DPM option computes the capacitances from polynomial coefficients supplied in the arrays. As for the other dual material options, multiplying factors are applied to the results of the calculations on the arrays. Eight data values are required to define each group of nodes, as follows:

```
DIM N#,#N,IN,Ti,AP1,F1,AP2,F2  $(basic format)
DPM N#,#N,IN,Ti,AC1,F1,AC2,F2  $(basic format)
DPM 30,3,5,70.0,A8,0.1,A6,K5   $ example 13
```

where:

- N# ..... Actual node number of the first node (integer)
- #N ..... Total number of nodes to be generated (integer)
- IN ..... Increment to be added to the current node number to form the node number of the next node (non-zero integer)
- Ti ..... Initial temperature of all nodes (floating point)
- AP1,AP2 .. References to arrays of points of T vs. C (An or smn.An form, where n is the array number)
- AC1,AC2 .. References to arrays of polynomial coefficients (An or smn.An form)
- F1,F2 .... Multiplying factors (floating point data value, or user constant reference of the form: XKm, Km, smn.XKm, or smn.Km, where m is the constant number)

Example 13 defines 3 nodes, numbers 30, 35, and 40, each made of two materials which have a temperature varying capacitance. Polynomial coefficients for the first material are supplied in array 8, and coefficients for the second material are supplied in array 6.

As for the DIV and DPV options, if one of the materials has a constant capacitance, the array reference for that material may be replaced by a floating point data value. The value so supplied or referenced will be used in place of the result of interpolations or polynomial evaluation on the corresponding array. This feature permits the following formats:

```
DIM N#,#N,IN,Ti,SUB,F1,AP2,F2    $(alt format)
DPM N#,#N,IN,Ti,SUB,F1,AC2,F2    $(alt format)
DIM N#,#N,IN,Ti,AP1,F1,SUB,F2    $(alt format)
DPM N#,#N,IN,Ti,AC1,F1,SUB,F2    $(alt format)
```

where SUB is a floating point data value.

BIV Option — The BIV options enables the user to define a node having a capacitance which is a function of both time and temperature. The format for such a node is similar to that for the SIV option, and requires four data values for each node, as follows:

```
BIV N#,Ti,AB,F                   $ (basic format)
BIV 55,70.0,A2,XK4               $ example 14
```

where:

N# ..... Actual node number (integer)
Ti ..... Initial temperature of the node (floating point)
AB ..... Reference to a bivariate array of temperature and time vs. capacitance. (An or smn.An form)
F ..... Multiplying factor (Floating point data value, or user constant reference of the form: XKm, Km, smn.XKm, or smn.Km, where m = constant number)

Examples 14 defines node 55 whose capacitance, as a function of time and temperature, is supplied in bivariate array number 2. Section 3.7.2 contains a description of the structure of a bivariate array. In this application, the X independent variable should be temperature, the Y independent variable should be mean time, and the Z dependent variable should be capacitance. When evaluating a node defined with the BIV option, the network solution routines will perform a bivariate interpolation on the referenced array. Using the current temperature of the node and the mean time for the current computation interval (control constant TIMEM) as the values of the independent variables.

3.8.2 Referencing Node Data

Each node is assigned an arbitrary submodel name and identification number by the user in the NODE DATA blocks. As each node is accepted by the preprocessor, it is renumbered in sequence. The user assigned identification number is called the *actual*, or node identification

number, and the preprocessor assigned sequence number is called the *relative* number. There is no restriction as to what order the four node types appear in the input file. After sorting by the preprocessor, they will be in diffusion, arithmetic, heater, and boundary order within each submodel. This is done for efficiency in solution routine execution and memory utilization.

Although the Fortran subroutines resulting from the translation of the SINDA operations block use the relative numbering system to access the node tables, it would be tedious and very error prone for the user to do so directly. Hence, the preprocessor maintains a table of actual vs. relative node numbers, and performs the conversion from the former to the latter whenever a temperature, capacitance or source table reference is encountered in the logic blocks. The node tables are referenced by the following forms:

```
smn.Tn = temperature
smn.Qn = source
smn.Cn = capacitance
```

where n is the actual (user specified identification) node number

By making the actual numbering system available outside the NODE DATA block, the preprocessor relieves the user of the burden of keeping track of the relative input order of the nodes. For example, the temperature of actual node number 6 in submodel SUBN is referenced by SUBN.T6, its capacitance by SUBN.C6, and its source by SUBN.Q6, regardless of the relative position of these quantities in their respective tables.

The smn.X(n) format is the "long" or complete reference form. This may be shortened to simply Xn when the submodel name is being supplied by an argument in the preceding header record. If the reference is to a T, Q or C outside the submodel specified in the header record, then the long form is required. The OPERATIONS DATA block is used for initialization and its logic relates to the entire model (as opposed to a single submodel) thus, the long form is the only one allowed in OPERATIONS DATA, unless DEFMOD is used.

3.8.3 Node Data Updating

The time- and temperature-dependent capacitance data inputs defined in the NODE DATA blocks are actually applied (that is, calculated and stored in the capacitance locations) at the end of the VARIABLES 1 logic block for that submodel. These updates are performed by calls to subroutine CVTEMP. The preprocessor normally inserts this call subsequent to any user-supplied logic in the VARIABLES 1 block. This gives the user an opportunity to update any user constants or arrays that may affect the node data calculations before they are performed. If user intervention is required *after* the CVTEMP call, the user may supply a CALL CVTEMP('smn') statement anywhere in the VARIABLES 1 input and it will not be overridden by a preprocessor-inserted call.

The SOURCE and CONDUCTOR DATA options, which will be described next, have equivalent time- and temperature-dependent update procedures using routines QVTIME, QVTEMP, GVTIME, and GVTEMP after VARIABLES 0 as well as VARIABLES 1. However, unlike those options, NODE DATA has no purely time-dependent options, and therefore has no analog to QVTIME and GVTIME—there is no "CVTIME."

3.9 SOURCE DATA

SOURCE DATA blocks provide the user with a convenient means for defining heat sources or sinks which are to be impressed on the nodes. Source must always be input in units of *energy per unit time*. A positive source means energy *input* to the node.

All SOURCE DATA records appear following a header record:

```
HEADER SOURCE DATA, smn
```

where `smn` is a submodel name.

The user should note that SOURCE DATA is really defining a node attribute, so there must be strict correspondence between the full node identification (submodel name plus node number) in all records within all the NODE DATA and SOURCE DATA blocks.

SOURCE DATA is unique among data blocks in that *numbered user constants (K and XK) may be used as part of the definition of the source term.* Furthermore, changes to the value of the user constant later in the execution will automatically affect the source term. However, there are two points to note when using such methods. First, no arithmetic expressions are allowed in combination with user constants in this block; only the Kn or XKn may appear as the input. Second, only the number n is important in the expression Kn or XKn, not the type implied by the use of K vs. XK. This cell is then used as a Fortran subroutine argument, meaning that the type (real or integer) is implied by the usage, not by the value. For this reason, either K or XK is valid at any time within this block; it is the user's responsibility to make sure that a real value was provided where a real value is needed, etc. In the formats that follow, the use of numbered user constants is indicated by "Km or smn.Km." Alternate use of XKm or smn.XKm is equally valid. Unless specifically stated to be a time-varying source (e.g., TVS), all source data dependent on user constants are updated at the end of VARIABLES 1 by QVTEMP (see Section 3.9.2)

3.9.1 Source Data Input

The SOURCE DATA input options and their associated 3-character codes are summarized in Table 3–7.

Standard Option (3 blanks) — This option impresses a constant source on a single node. The input format is as follows:

```
N#,Q                              $ (basic format)
4,1.8,6,K3 or smn.K3              $ (example 1)
```

where:

N# ..... Actual number of a node defined in a NODE DATA block (integer).

Q ..... The value of the source (floating point data value, or user constant reference of the form: XKm, Km, smn.XKm, or smn.Km, where m is the actual number of a floating point user constant)

Table 3-7 Summary of SOURCE DATA Input Options

OPTION (CODE)	DESCRIPTION
3 blanks	To impress a constant heat source on a single node
GEN	To impress the same heat source on several nodes
SIV	To impress a temperature-varying heat source on a node
SIT†	To impress a time-varying heat source on a node
DIT†	To impress the sum of two time-varying heat sources on a node
TVS	To impress a time-varying heat source on a node
TVD	To impress the sum of two time-varying heat sources on a node
DTV	To impress the sum of a time-varying source and a temperature-varying source on a node
CYC†	To impress a cyclic time-dependent source on a node
PER	To impress a cyclic time-dependent source on a node

† Beginning with SINDA '85, the SIT, DIT, and CYC options have been replaced with the TVS, TVD, and PER options respectively. The obsolete options are included only for processing old SINDA files.

Example 1 will cause a heat rate of 1.8 to be impressed on node number 4, and the floating point heat rate stored in user constant number 3 to be impressed on node number 6.

In order to enforce the correlation between NODE DATA and SOURCE DATA, the form smn.N (i.e., a reference to a node in another submodel) is not allowed in any SOURCE DATA format.

GEN Option — The GEN options allows the user to impress the same heat source on a group of several nodes. The format is as follows:

```
GEN N#,#N,IN,Q                    $(basic format)
GEN 6,3,1,4.3                     $(example 2)
```

where:

N# ..... Actual number of the first node to receive the source (integer)
#N ..... The total number of nodes to receive this source (integer)
IN ..... The increment to be added to the current node number to form the next node number (non-zero integer)
Q ..... The value of the source (floating point data value, or user constant reference of the form: XKm, Km, smn.XKm, or smn.Km, where m is the actual number of a floating point user constant)

Example 2 will cause a heat rate of 4.3 to be impressed on nodes 6, 7, and 8.

Automated Variable Source Options — The automated options provide for operations which will automatically evaluate the current value of a heat source which varies with time or temperature (or both), and place this value in the appropriate source location.

All options cause the source to be evaluated by performing linear interpolation on arrays of values which represent points on a curve of time or temperature vs. heat rate. If the heat source is temperature dependent, the values are entered in the ARRAY DATA block in the following doublet array format:

$$x_1, y_1, \ x_2, y_2, \ \ldots \ x_i, y_i, \ \ldots \ x_n, y_n$$

where:

- y_i ..... Dependent variable - heat rate
- x_i ..... Independent variable - temperature; x_i (i = 1, 2, ..., n) is strictly increasing with i
- x_i, y_i ..... Floating point numbers

and are referenced in the SOURCE DATA block by using the form: Am or smn.Am, where m is the array identification number.

For temperature-dependent sources, the value of the independent variable used for interpolation will be the current temperature of the node receiving the source. If the value of the independent variable is less than x1, then y1 will be used in lieu of performing an extrapolation. Similarly, if the value is greater than xn, then yn will be used. To provide further versatility, the value of the heat rate found from interpolation will be multiplied by a factor before being entered in the source table. This multiplying factor can assume any significance that is convenient for the user. If the multiplying factor has any units, the product of these units and those of the dependent variables in the array must be units of heat rate.

SIV Option — The SIV option allows the user to specify a temperature varying source. The record format is as follows:

```
SIV N#,A,F                    $(basic format)
SIV 8,A4,1.0                  $(example 3)
```

where:

- N# ..... Actual number of the node which will receive this source (integer)
- A ..... Reference to doublet array of temperature vs. heat rate, of the form: An or smn.An, where n is the actual array number
- F ..... Multiplying factor (floating point data value, or a reference to a floating point user constant of the form: XKm, Km, smn.XKm, or smn.Km, where m is the actual number of a floating point user constant Km or smn.Km, and where m is the actual constant number)

The following discussion, and example 3 above, illustrates the use of the SIV option.

Assume that the heat rate defined by the graph in Figure 3–4 is to be impressed on node number 8. This graph can be represented by the following doublet array, which would be entered in an ARRAY DATA block:

```
4 = 0.0,0.0,10.0,0.1,20.0,0.05,30.0,0.05,40.0,0.0
```

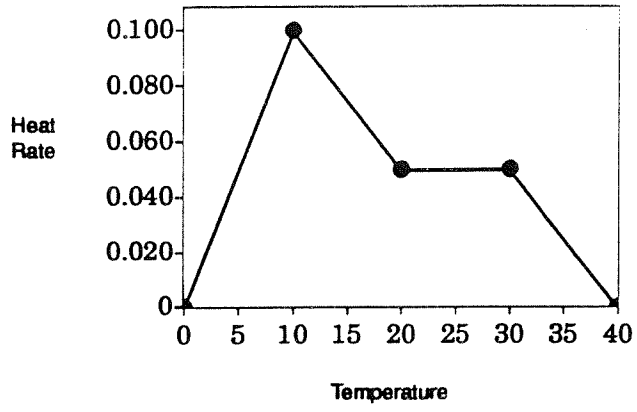


Figure 3–4 Temperature-Varying Heat Rate

NOTE: These doublet arrays are not allowed in TARRAY DATA blocks. The required heat source is then defined in the source data by the record shown in example 3. Since the referenced array (array number 4) does not need to be scaled or modified in any way, the multiplying factor is merely given as the floating point data value 1.0.

TVS Option — The TVS option allows the user to apply a time varying source to a node. The difference between the TVS and SIV options is that the array referenced with the TVS option may be entered in either the ARRAY DATA or TARRAY DATA blocks and must be in the singlet form. There is a restriction that both the time array and q-arrays must be in the same data block. The ARRAY or TARRAY DATA formats are as follows:

```
nt      =  t1,t2,t3 - - - - tn
nq      =  q1,q2,q3 - - - - qn
```

where:

`nt` ..... Time array number
`ti` ..... Time values (independent variable, values strictly increasing with i)
`nq` ..... Heat rate array number
`qi` ..... Heat rate values (dependent variable)
`ti,qi` .... Floating point numbers

The TVS SOURCE DATA input format is as follows:

```
TVS N#, At, Aq, F          $(basic format)
TVS 16,A1,A9,K13           $(example 4)
```

where:

N#	Actual number of the node which will receive this source (integer)
At	Reference to singlet time array of the form: An or smn.An, where n is the array number
Aq	Reference to singlet heat source array same form
F	Multiplying factor (floating point data value, or a reference to a floating point user constant of the form: XKm, Km, smn.XKm, or smn.Km, where m is the actual number of a floating point user constant)

Example 4 causes the results of interpolating on arrays 1 and 9, (using the mean time for the computation interval (TIMEM) as the independent variable) multiplied by the value in user constant 13 to be inserted in the source location for node number 16, Q16.

TVD Option — The TVD option is used to specify a heat source which is the sum of two separate, time-varying sources. The user must supply two singlet time vs. heat rate arrays, in either an ARRAY or TARRAY DATA block, and a multiplying factor for each of the two sources, as follows:

```
TVD N#,At,Aq,FA,Bt,Bq,FB $(basic format)
TVD 3,A1,A5,K2,A1,A6,K4  $(example 5)
```

where:

N#	Actual number of the node which will receive this source (integer)
At,Aq, Bt,Bq	References to singlet arrays of time vs. heat rate for source A and source B, respectively of the form: An or smn.An, where n is the array number
FA,FB	Multiplying factors to be applied to the results of interpolation. (floating point data values, or references to floating point user constants, of the form: XKm, Km, smn.XKm, or smn.Km, where m is the actual number of a floating point user constant)

Example 5 defines a source to be impressed on node 3, which will be evaluated by taking the sum of the interpolation result from array 5 times the value of user constant 2, and the interpolation result from array 6 times the value of user constant 4. Both interpolations use time array 1 for the independent variable data.

DTV Option — This option enables the user to specify a source which is the sum of a time varying source and a temperature varying source. The DTV option requires separate doublet arrays for time vs. heat rate and temperature vs. heat rate, and represents the summation Q = f(t) + g(T). The record format is as follows:

```
DTV N#,At,Ft,AT,FT              $(basic format)
DTV 26,A14,K1,A16,K3            $(example 6)
```

where:

N# ..... Actual number of the node which will receive this source (integer)

At ..... Reference to a doublet array of time vs. heat rate, (form: An or smn.An, where n is the array number)

Ft ..... Multiplying factor to be applied to the result of interpolation on array At (floating point data value or a user constant reference of the form: Km or smn.Km, where m = actual constant number)

AT ..... Reference to a doublet array of temperature vs. heat rate, form: An or smn.An, where n is the array number

FT ..... Multiplying factor to be applied to the result of interpolation on array AT (floating point data value, or a reference to a floating point user constant of the form: XKm, Km, smn.XKm, or smn.Km, where m is the actual number of a floating point user constant)

Example 6 defines a source, to be impressed on node 26, which will be evaluated at each iteration, as the sum of a time varying component (calculated from array 14 and user constant 1) and a temperature varying component (calculated from array 16 and constant 3).

The DTV option may be used to define a source which is the sum of (1) a time varying source and a constant source, or (2) a constant source and a temperature varying source. (Case 1 is equivalent in effect to the alternate form for the TVD option, but is also made available to the DTV option for the sake of consistency.) Case 1 is accomplished by replacing the array reference, At, with a floating point data value. Case 2 is accomplished by replacing the array reference, AT, with a floating point data value. In either case, the value of the constant source is taken as the product of the data value, or referenced constant, and its associated multiplying factor. The alternate formats are as follows:

```
DTV N#,SUB,Ft,AT,FT                $(case 1)
DTV N#,At,Ft,SUB,FT                $(case 2)
```

where SUB is a floating point data value.

PER Option — The PER option in the SOURCE DATA block provides for automated cyclic interpolation of time varying heat rates. The record format is as follows:

```
PER N#,At,Aq,F,D,P
PER 12,A1,A5,1.0,.5,1.0
PER 13,A1,A5,1.0,.75,1.0 $(example 7)
PER 14,A1,A5,1.0,0.,1.0
PER 15,A1,A5,1.0,.25,1.0
```

where:

N#	Actual node number (positive integer)
At,Aq	Reference to singlet arrays of mean time vs. heat rate points (integer) The time array must start at zero. Reference form is An or smn.An where n is the actual array number
F	Multiplying factor (floating point data value, or reference to a user constant)
D	Phase displacement as a decimal fraction of the total period. (Must be a floating point literal with a value between 0.0 and 1.0. A user constant reference is not allowed.)
P	Period specified in time units (floating point data value or reference to a user constant). P is usually the last time point in the At array, but not restricted to this value.

If P is less than the last At array value, any values between P and the last array value are not used. If P is greater than the last array time value, the heating value used for the time period between the last array time value and P is constant and equal to the heating rate input for the last array time value.

Example 7 defines cyclic sources to be impressed on nodes 12, 13, 14, and 15 of a cylindrical spacecraft rotating clockwise normal to the sun at one revolution per hour. The heating rates to the four small surfaces shown in Figure 3–5 can be obtained as follows.

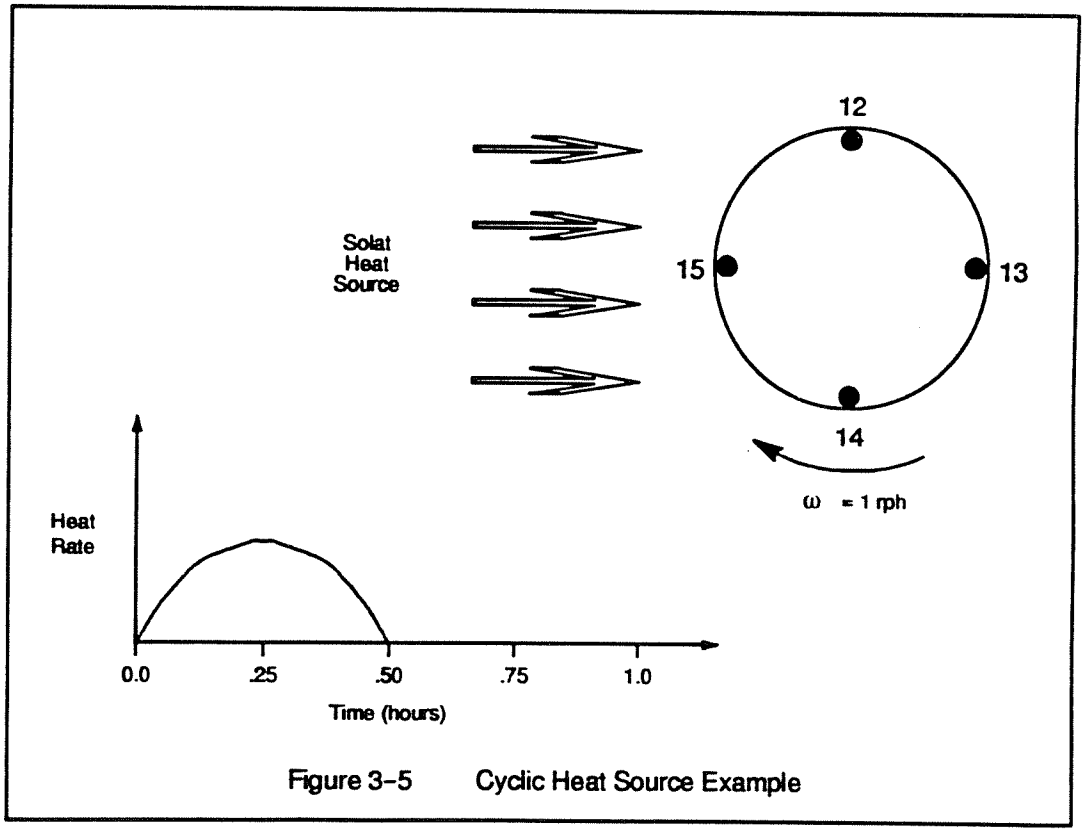


Figure 3–5 Cyclic Heat Source Example

The heating data is entered in an ARRAY DATA or TARRAY DATA block in the usual tabular form. For this case, data is assumed to be entered into arrays 1 and 5. With the cylindrical spacecraft rotating clockwise and with the above orientation of each surface with respect to the sun at time zero, the phase displacements are as follows:

NODE	PHASE DISPLACEMENT
14	0.0
15	0.25
12	0.50
13	0.75

The period is 1.0 hour.

3.9.2 Source Data Updating

All references to source data in the logic blocks use the forms: Qn or smn.Qn where smn is a submodel name and n is the actual node number. As usual, the "smn." prefix is required only in the OPERATIONS or SUBROUTINE DATA blocks and for cross-references to a node outside the submodel stated in the header or DEFMOD record currently in effect. With this accessing capability, the user may update source data values at will with statements in any logic block.

The time and temperature dependent SOURCE DATA inputs defined in the SOURCE DATA blocks are actually applied (that is, stored in the source locations) at the end of VARIABLES 0 and VARIABLES 1 for time and temperature dependent operations, respectively. The DTV option (time and temperature dependent) is done in VARIABLES 1 prior to each temperature iteration. Three-blanks and GEN options that are dependent on user constants are updated after VARIABLES 1.

These updates are done at the end of the VARIABLES 0 and VARIABLES 1 subroutines through calls to program subroutines QVTIME and QVTEMP. The preprocessor normally supplies these calls subsequent to any user-supplied logic in the VARIABLES 0 or VARIABLES 1 blocks. This gives the user an opportunity to update any user constants or arrays that may affect the SOURCE DATA calculations before they are performed. If user intervention is required *after* the QVTIME/QVTEMP calls, the user may supply a "CALL QVTIME" statement anywhere in the VARIABLES 0 input and it will not be overridden by a preprocessor-supplied call. This is also true for QVTEMP in VARIABLES 1.

To conform with previous SINDA practice, ***all time and temperature dependent source terms are set to zero (or their "constant" SOURCE DATA value) at the beginning of VARIABLES 0.*** All constant value sources input in the SOURCE DATA block are reset to their initial values at the beginning of VARIABLES 0. Sources will only be zeroed at that point if no source term had been input or if the constant source value were zero. *In steady-state runs, where VARIABLES 0 is called only once, source data values are reset to their post-VARIABLES 0 (and QVTIME) values at the start of each iteration step.*

Heat transfer ties to fluid submodels (described later) affect the nodal Q-source value. The effect of these ties appears after the VARIABLES 0 (and QVTIME) updates, but before the VARIABLES 1 (and QVTEMP) updates *in all solution routines.* FLUINT ties are therefore treated like time-dependent sources in transient solutions. In steady-state solutions, on the other hand, tied lumps are treated like boundary nodes from the nodes' perspectives.

3.10 CONDUCTOR DATA

Conductors are SINDA network elements that describe how energy is to be transported between two nodes. Conductors are heat transfer paths. Two basic types of conductors may be defined by the user in CONDUCTOR DATA blocks: *linear* and *radiation*.

Linear Conductors - The conductance of a linear conductor is input in units of *energy per unit time per unit degree**, and the heat rate through such a conductor is calculated in the network solution routines as:

$$Q = G \bullet (T_i - T)_j$$

where:

Q ..... Heat rate (energy/time)
G ..... Linear conductance
T ..... Temperature of endpoint nodes i and j

Several types of physical heat transfer mechanisms can be modeled as linear conductors. For heat transfer by conduction, the conductance should be computed as:

$$G = \frac{k \bullet A}{L}$$

where:

k ..... Thermal conductivity (energy/length-time-deg)
A ..... Cross-sectional area of the conduction path (length^2)
L ..... Length of the conduction path (length)

For heat transfer by convection, the conductance should be computed as:

$$G = h \bullet A$$

where:

h ..... Convective film coefficient (energy/length^2-time-deg)
A ..... Surface area (length^2)

For heat transfer by mass flow, the conductance should be computed as:

$$G = \dot{m} \bullet Cp$$

where:

$\dot{m}$ ..... Mass flow rate (mass/time)
Cp ..... Specific heat of the flowing material (energy/mass-deg)

---

* Temperature can be in degrees F, C, R, or K as long as all data (including mass, length, and time) contributing to the G-value constitute a consistent set. Consistency must also exist among all the capacitance and temperature values in the set of active submodels. The tac and tcon arguments on HEADER records and the FAC macro are provided for this. The temperatures must also be consistent with the value ABSZRO. When fluid models are active, the conventions implied by the UID control flag must also be followed.

Radiation Conductors — The conductance of a radiation conductor is input in units* of *energy per unit time per degree*^{4}, and the heat rate through such a conductor is calculated in the network solution routines as:

$$Q = G \bullet (T_i^4 - T_j^4)$$

where:

Q ..... Heat rate (energy/time)
G ..... Radiation conductance
T ..... *Absolute* temperature of endpoint nodes i and j

The value input as the conductance of a radiation conductor should be computed as:

$$G_{input} = \sigma \bullet \mathcal{F} \bullet A$$

where:

σ ..... Stefan-Boltzmann constant (energy/length^2-time-deg^4, e.g., 0.1714E-8 BTU/ft^2-hr-R^4 or 5.667E-8 W/m^2-K^4)
$\mathcal{F}$ ..... Gray-body factor
A ..... Surface area (length^2)

Preliminary Input Rules — All conductor input options require that the pair of nodes to which a conductor is attached be specified by their actual node numbers. The user should recall that the negative sign preceding boundary node numbers on node data cards is *not* part of the actual node number. On conductor data cards, negative signs prefixing node numbers are not flagged as errors because they are used to signify one way conductors (see below).

If *both* nodes connected by a conductor are in the submodel named in the preceding HEADER CONDUCTOR DATA record, just the actual node number (N#) will suffice. Since inter-submodel heat transfer is allowed, the form smn.N# is available when one of the pair is in another submodel. The submodel format for upstream nodes in one way conductors is smn.-N#. Note that the second submodel name is not echoed in the preprocessor printback for conductor generation options for reasons of internal execution economy.

A conductor may also appear with both node numbers identified with submodels outside the conductor's submodel. This allows the user to define one or more heat transfer paths between submodels to be changed or eliminated simply by using the BUILD macroinstruction. The user should keep in mind that the physical constituents of his model are primarily diffusion and arithmetic nodes, secondarily conductors and boundary nodes.

* In addition to the units consistency required for linear conductors, radiation conductors must also be consistent with the SIGMA value specified in CONTROL Data.

One-Way Conductors — Historically, to facilitate the modeling of fluid loops without a fluid submodel, SINDA allows any conductor to be specified as a *one-way conductor*. One-way conductors permit the modeling of heat transport by fluid (mass) flow, in which case their conductances are always a function of $\dot{m} \cdot Cp$. Such a conductor is defined by prefixing the node number of one of the adjoining nodes with a minus sign. The node so designated* will not be allowed to lose or gain heat through the conductor, even though the temperature of the node will be used to calculate a heat flow. In other words, the solution subroutines will compute the heat transferred through a one-way conductor as though it were an ordinary conductor. This heat will not, however, be allowed to enter (or leave) the node prefixed by the minus sign; it will be allowed to leave (or enter) the unsigned node only.

In other words, the "downstream" (positive) node solution includes the effects of the "upstream" (negative-flagged) node, *but the upstream node is completely unaware of the existence of the downstream node. This has significant modeling repercussions beyond simple fluid flow modeling, which is better accommodated using fluid submodels.* For example, one-way radiation conductors are perfectly valid.

One important use of one-way conductors is to reduce model sizes by exploiting symmetries. Dual, opposing one-way conductors (see Figure 4–17) can be used in a manner analogous to the FLUINT path and tie DUP options described later. For example, if a model contains N identical subnetworks all attached to the same main node, then one one-way conductor of value G could be placed from the main node to the "duplicated" subnetwork node, and a second of value N*G would be placed from the subnetwork node to the main node. The main node would then "see" N subnetwork nodes, whereas the subnetwork node would only "see" one main node.

N need not be an integer. For example, consider an insulated box with X square foot of surface area contained M widgets. A model of one square foot of insulation could be developed independent of a model of one widget, and the two could later be combined (perhaps as separate submodels) using dual one-way conductors. The widget model would "see" X/M square feet of insulation, whereas the insulation model would "see" M/X widgets. No further modifications would be necessary to produce a model of the combined system, assuming that the unit systems matched or that one could be converted using FAC macrocommands. Such interpretation and usage is also handy when reducing a detailed component model to a simpler, faster system–level model, as illustrated by Sample Problem G in Appendix F.

The use of one-way conductors has effects on the model energy balance, and on QMAP and NODMAP output files. In QMAP and NODMAP files, downstream nodes do not appear in the listings for the upstream nodes since they have no effect on those nodes. Because of this lack of one-to-one correspondence, energy can appear to be created or destroyed at these locations, which can disrupt the system-level energy balance calculation required for steady-state convergence (as governed by the EBALSA control constant). In this event, SINDA will converge, but will print a warning that the net energy flows are unbalanced. This should not be misinterpreted as signalling an incorrect answer if one–way conductors are used. However, large conductors can also cause this message, in which case the program is attempting to warn the user that little to no progress is being made toward a solution, and that either tighter convergence criteria or changes/corrections to the model are needed.

---

* If fluid flows from node A to node B, then node A (the upstream node) should be flagged with a minus sign to signal a one–way conductor. If the upstream node is outside of the current submodel, the correct designation is: smn.–N#. *NOTE: One–way conductor representations of fluid flow are completely unrelated to fluid submodels.*

3.10.1 Conductor Data Input

All Conductor data records appear following a header record:

```
HEADER CONDUCTOR DATA, smn [,fac]
```

where:

`smn`	a submodel name
`fac`	conductance multiplying factor
`[ ]`	denotes optional argument

The following general format rules apply for all conductor input options:

1. Linear conductors must have a positive conductor number.
2. Radiation conductors must have a negative conductor number (analogous to boundary node designations, the negative sign is used purely as a signal and has no other meaning).
3. Multiple conductors, or groups of same-valued conductors, may be defined on a single input record using the Standard (3 blanks) option code. All other option codes appear only once per record, together with their data values.
4. Conductor identification numbers may consist of up to 6 digits (e.g., 999999).

Table 3–8 summarizes the available conductor data input options. Like all other model parameters, conductance values may be referenced and changed in the logic blocks. Hence, a "constant" value input in conductor data need not remain constant during the entire problem solution.

Standard Option (3 blanks card code) — The simplest conductor data card utilizes the 3 blank option and is formatted as follows:

```
G#,NA,NB,G                    $(basic format)
-4,7,9,3.5E-13                $(example 1)
18,31,POD2.8,1.8              $(example 2)
```

where:

`G#`	Actual conductor number (integer)
`NA,NB`	Actual node numbers of the nodes to which this conductor is connected (integer data value or smn.integer)
`G`	Conductance (floating point data value)

Example 1 defines conductor number 4 as a radiation conductor connected between nodes 7 and 9 and having a conductance value of 3.5×10^{-13}. Example 2 defines conductor number 18 as a linear conductor connected between nodes 31 and node 8 of submodel POD2, having a conductance value of 1.8.

Table 3-8 Summary of CONDUCTOR Data Input Options

OPTION (CODE)	DESCRIPTION
3 blanks	To input single conductors
CAL†	To input single conductors where G is a strict function of 4 factors
GEN	To generate several conductors with the same conductance
SIV SPV	To input single temperature-varying conductors. SIV uses array interpolation, SPV uses a polynomial function of temperature
SIM SPM	Same as SIV and SPV for groups of conductors
DIV DPV	To input single conductors that model two materials with different temperature-varying conductances. DIV uses two arrays of G vs. T, and DPV uses polynomial functions of T.
DIM DPM	Same as DIV and DPV for groups of conductors
BIV	To input a single conductor that is both time- and temperature-varying. The conductance is found by interpolation using a bivariate array.
PIV PIM	To input conductors where the conductance is a function of temperature and a time-independent variable. PIV is for single conductors, PIM is for multiple conductors
TVS	To input a time-varying conductor
PER	To input a cyclic time-varying conductor

† Beginning with SINDA '85, floating point arithmetic (+,-,*,/) in data fields makes the CAL option obsolete. The CAL option is available but not documented.

Since a conductor has a single attribute, conductance, regardless of its type, a single mathematical conductor may be used to represent any number of physical conductors. In this way, *any number of conductors having the same conductance may be assigned to the same conductor number*, as follows:

```
G#,NA1,NB1,NA2,NB2, ...,NAx,NBx,G          $(alt format)
11,2,3,4,7,3.8                             $(example 3)
14,7,6,-9,8,2.1                            $(example 4)
```

where `NAi,NBi` (i = 1, 2, ... x) are node numbers of adjoining node pairs (integer or smn.integer). This usage is an important "trick" which can be used to significantly reduce both inputs and logic: manipulation of a single conductor value can change the conductance between many pairs of otherwise unrelated nodes.

Example 3 defines linear conductor number 11 as having a value of 3.8; it is connected between nodes 2 and 3, and *also* between nodes 4 and 7. Even though the conductor number for these two conductors is the same, no *physical* connection between node (2 or 3) and node (4 or 7) has been explicitly or implicitly established. Example 4 defines a linear conductor, having a value of 2.1, connected between nodes 7 and 6, and connected as a one-way conductor between nodes 9 and 8. In the latter case, heat will flow to and from node 8, but will be prevented from entering or leaving node 9 (through this conduction path).

The inclusion of more than a single (NA, NB) node pair for a given conductor is allowed only with the standard (3 blanks) conductor data input option, and with the GEN option described below (when IG is zero—see footnote). Since the total number of data values required to define a conductor (using the alternate format) is variable, the general requirement that each conductor be defined on a single card is relaxed to allow the definition to be split across several cards. In this manner, an unlimited number of like-valued conductors may be modeled by one conductor.

GEN Option — The GEN option is used to generate and input a group of conductors each having the same conductance value. This option requires eight data values for each group of similar conductors, as follows:

```
GEN G#,#G,IG,NA,INA,NB,INB,G      $(basic format)
GEN -5,3,1,10,1,20,1,4.8E-12      $(example 7)
GEN 30,3,0,-40,1,50,-1,2.6        $(example 8)
```

where:

G# ..... Actual number of the first conductor to be generated (integer)

#G ..... Total number of conductors to be generated (integer)

IG ..... Increment to be added to the current conductor number to form the conductor number of the next conductor (integer)

NA,NB .... Actual node numbers of the pair of nodes connected by the first conductor (integer or smn.integer)

INA,INB .. Increments to be added to NA and NB, respectively, to arrive at the node numbers for the pair of nodes connected by the next conductor (integer)

G ..... Conductance of all conductors (floating point)

A negative sign preceding the G# causes all conductors to be defined as radiation conductors. The negative sign, however, is never part of the conductor number in the arithmetic sense. Therefore, the conductor numbers generated in example 7 would be 5, 6 and 7 (not -5, -4 and -3), and all three would be radiation conductors. Similarly, a negative sign preceding NA or NB would define the group of conductors as being the one-way type. This negative sign, too, is never part of the node number in the arithmetic sense. Therefore, the NA, NB node pairs generated in example 8 would be (40, 50), (41, 49), and (42, 48). The increment values, IG, INA, and INB, may assume any integer value (positive, negative, or zero)*. Example 7 is equivalent to the following cards prepared under the standard (3-blanks) option:

```
-5,10,20,4.8E-12
-6,11,21,4.8E-12
-7,12,22,4.8E-12
```

* When IG is zero, a single conductor will be connected between each generated NA,NB pair. The standard option, which permits several such pairs to be specified for a single conductor, is another way to create a multiply-defined conductor.

Example 8 is equivalent to the following record, which uses the alternate form of the standard (3 blanks) option:

```
30,-40,50,-41,49,-42,48,2.6
```

A "fan" of 5 like-valued conductors may be generated as follows:

```
GEN 51,5,1,22,0,31,10,6.12*.025/16.2     $ example 9
```

The zero entry for INA generates node pairs as follows: (22,31), (22,41), (22,51), (22,61), (22,71). Note the arithmetic in the G value field.*

Automated Variable Conductance Options — As stated earlier, the network solution routines call for the execution of the operations in the VARIABLES 1 block prior to performing each iteration on the heat transfer equations. Using this feature, the user could get a temperature-dependent conductance by specifying VARIABLES 1 operations which would insert a new conductance value in the conductance table. This new conductance could be calculated using any convenient algorithm. This is probably the most efficient way, from a computer time standpoint, for this operation but it tends to be rather cumbersome.

The automated options cause the current value of variable conductors to be calculated within the network solution routines at the end of the VARIABLES 1 operations, and therefore relieve the user of the task of preparing and inputting conductance computation algorithms. As specified by the user, one of two standard methods is used to calculate the current value of a variable conductor: (1) interpolation, and (2) polynomial evaluation. For interpolation, the user must input, in the ARRAY DATA block, a bivariate array of (temperature, conductance) ordered pairs representing points on the curve of T vs. G. The conductance is then calculated by performing linear interpolation on this array using temperature as the independent variable. For polynomial evaluation, the user must input an array of coefficients (P_0, P_1, P_2, ... P_n) in the ARRAY DATA block. The conductance would then be calculated by evaluating the n^{th} order polynomial in T (i.e., $P_0+P_1*T+P_2*T**2 ... +P_n*T**n$). To provide further versatility, the value of the temperature, T, used as the independent variable in both of the above methods, may be taken as the average of the temperatures of the two nodes to which the conductor is connected, or it may be set equal to the temperature of only one of the nodes. This feature is explained along with the options to which it applies.

To enable the same array of points or coefficients to be used for all conductors passing through the same type of material, all options cause the computed conductance to be multiplied by a factor before being inserted in memory. In this way, for example, the array can be used to compute conductivity, k, and the multiplying factor can be used for A/L in the case of linear conductors or, emissivity, ε, and A, respectively, in the case of radiation conductors. In another application, the multiplying factor could be used to scale the conductance to some set of consistent units.

---

* The eleven-data-value GEN option has been made obsolete by the Fortran arithmetic feature of SINDA/FLUINT. It is available but not documented.

All of the automated options require a multiplying factor which may be input as a floating point data value, or as a user constant of the form: XKn, Kn, smn.XKn, or smn.Kn, where n is the actual number of a user constant. If the latter form is used, the referenced constant must be a floating point value. Note that the user may access and change this constant, and thus the conductor value algorithm, at any point in the solution process.

The interpolation options (except BIV*) require a doublet array of temperature vs. conductance. (Conductance is used loosely in this context. The array could contain, for example, values of temperature vs. conductivity, as long as the associated multiplying factor had units of area/length.) The array must contain an even number of floating point data values, as follows:

$$T_1, G_1, T_2, G_2, \ \ldots, \ T_j, G_j, \ \ldots, T_n, G_n$$

The temperature values, T_j, must be strictly increasing with j. If the value of the effective temperature (average or single node) at some time is less than T_1, then G_1 will be used in lieu of performing an extrapolation; if the temperature is greater than T_n, then G_n will be used.

The polynomial options require that a singlet array of coefficients be input in the ARRAY DATA block. The array must consist of n+1 floating point values, where n is the order of the polynomial. The coefficients should be entered in the array in the order of increasing powers of temperature (e.g., the first data value is the constant term, P_0, and the $(n+1)^{th}$ data value is the coefficient, P_n, of T**n).

For both classes of options, the arrays are referenced by using the form: An or smn.An, where n is the array number.

SIV and SPV Options - These options allow the user to define single conductors having temperature varying conductance. The SIV option assumes the referenced array contains points of T vs. G and is to be used for interpolation, whereas the SPV option assumes that the array contains polynomial coefficients. Both options require five data values for each conductor as follows:

```
SIV G#,NA,NB,AP,F                $(basic format)
SPV G#,NA,NB,AC,F                $(basic format)
SIV -12,24,-25,A3,-0.0001        $(example 10)
SPV 4,5,6,A2,K11                 $(example 11)
SIV 4,5,6,A3,0.0001              $(example 12)
```

* The BIV option requires a bivariate array of time and temperature vs. conductance

where:

G#	Actual conductor number (integer)
NA,NB	Actual node numbers of the pair of nodes to which the conductor is connected (integer or smn.integer)
AP,AC	A reference to an array of points, or coefficients, respectively, of the form: An or smn.An, where: n is the array number
F	Multiplying factor, input as a positive floating point data value, or a user constant reference of the form: XKm, Km, smn.XKm, or smn.Km, where m is the actual number or a user constant (arithmetic expression not allowed). The constant so referenced must have been input as a floating point data value in the USER DATA block.

As for all other options, this type of conductor may be specified as a one-way conductor by prefixing NA or NB with a minus sign.

Normally, the average temperature of the two adjoining nodes (i.e., $(T_{NA} + T_{NB})/2$) will be used as the independent value for interpolation or polynomial evaluation. If the user wishes to use the temperature of one of the nodes only then this node is identified by placing its number as the NA data value (not the NB value), and the single temperature option is indicated by prefixing the multiplying factor or user constant with a minus sign.

Since the negative serves only as a flag, the multiplying factor will always be interpreted as a positive quantity. Example 10 illustrates the correct input for a radiation conductor connected between nodes 24 and 25 but conducting no heat to or from node 25, wherein the temperature of node 24 will be used as the independent variable for interpolating on the T vs. G points supplied in array number 3, with the result of the interpolation being multiplied by 0.0001 (NOT -0.0001!) prior to insertion into memory. The above example, while unrealistic, encompasses all three methods for using minus signs as flags to establish conductor definition and evaluation options.

Examples 11 and 12 are discussed in detail in the following paragraphs.

Consider the thermal network shown in Figure 3–6. The conductance of the conductor is defined by the following equation:

$$G = (0.0001) \times (0.005 \times T^2 + 1.0)$$

where T is the average temperature of adjoining nodes.

Example 11 illustrates the correct method of defining this conductor using the SPV option. The required coefficients of the polynomial would be entered in the ARRAY DATA block with the following record:

```
2 = 1.0,0.0,0.005
```

The required multiplying factor would be entered in the USER DATA block with the following record:

```
11 = 0.0001
```

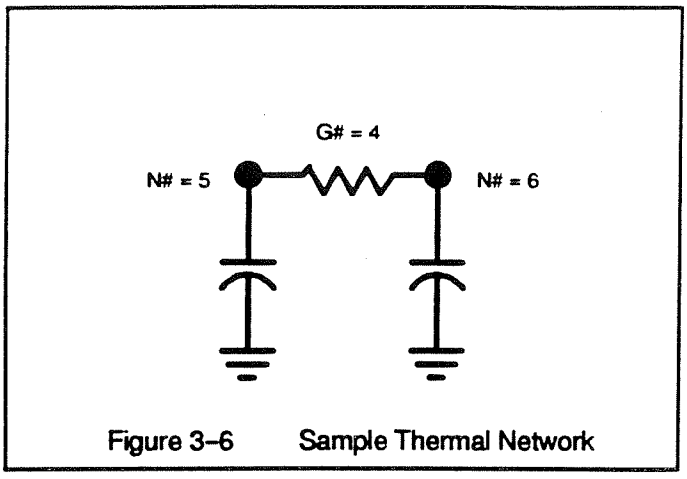


Figure 3–6 Sample Thermal Network

An alternate method of defining this conductor is illustrated in Example 12. This example uses the SIV option and thus requires an array of T vs. G points for interpolation. The array is developed by approximating the T vs. G curve with a series of straight lines as shown in Figure 3–7. The points which define this approximation are entered in the ARRAY DATA block as follows:

```
3 = 0.0,1.0,10.0,1.5,20.0,3.0,40.0,9.0
```

The multiplying factor is supplied directly on the conductor data card as the floating point data value, 0.0001. A user constant reference, such as K11, could have been used alternatively.

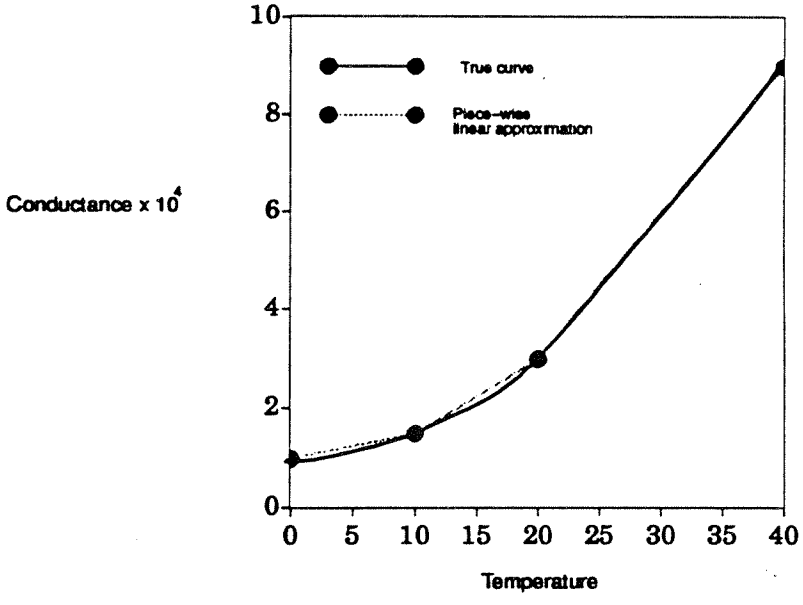


Figure 3–7 Curve of Temperature-Varying Conductance

A common usage of these options is to input an array of k vs. T (i.e., conductivity versus temperature) *once* for each material used in the model, and then to specify appropriate A/L (heat transfer area over length) of each linear conductance as the multiplying factor in each conductor declaration.

SIM and SPM Options — These two options combine the features of the SIV and SPV options with those of the GEN option. That is, they allow the user to generate and input a group of conductors all having the same temperature varying conductance. The SIM option generates conductors whose conductance will be evaluated by interpolation, and the SPM option generates conductors whose capacitance will be evaluated from a polynomial. Nine data values are required to define each group of conductors as follows:

```
SIM G#,#G,IG,NA,INA,NB,INB,AP,F     $(basic format)
SPM G#,#G,IG,NA,INA,NB,INB,AC,F     $(basic format)
SIM 10,3,1,5,1,6,1,A3,K11           $(example 13)
```

where:

- `G#` ..... Actual conductor number of first conductor to be generated (integer)
- `#G` ..... Total number of conductors to be generated (integer)
- `IG` ..... Increment to be added to the current conductor number to form the conductor number of the next conductor to be generated (non-zero integer)
- `NA,NB` .... Actual node numbers of the pair of nodes to which the first conductor is connected (integer or smn.integer)
- `INA,INB` .. Increments to be added to NA and NB, respectively, to form the node numbers for the pair of nodes connected by the next conductor (integer)
- `AP,AC` .... A reference to an array of T vs. G points, or polynomial coefficients, respectively, of the form: An or smn.An where n is the array number
- `F` ..... Multiplying factor input as a floating point data value, or a user constant reference of the form: XKm, Km, smn.XKm, or smn.Km, where m is the constant number of a floating point constant

The conductors defined by Example 13 are equivalent to those defined by the following:

```
SIV 10,5,6,A3,K11
SIV 11,6,7,A3,K11
SIV 12,7,8,A3,K11
```

As for all other options, a negative sign prefixing the G# establishes the entire group as radiation conductors; a negative sign prefixing NA or NB defines a group of one-way conductors: a negative sign preceding F (where F is input as a data value, not a user constant reference) establishes the temperature of node NA as the value of the independent variable to be used for interpolation or polynomial evaluation. Negative signs preceding G#, NA, or NB serve as flags to establish the type of conductors, in accordance with the usual conventions. Negative signs preceding IG, INA, or INB, however, indicate that the data value is, indeed, a negative arithmetic quantity. IG must be non-zero because multiply-defined conductors are permitted only with the standard (3 blanks) and GEN options.

DIV and DPV Options — These options allow the user to define a conductor which passes through two materials, each having a different temperature varying conductance. The DIV option uses the interpolation method to evaluate the conductance, whereas the DPM option uses a polynomial. Both options permit one material to have a constant conductance. Both options require seven data values for each conductor as follows:

```
DIV G#,NA,NB,APA,FA,APB,FB          $(basic format)
DPV G#,NA,NB,ACA,FA,ACB,FB          $(basic format)
DPV 10,20,30,A8,K9,A14,0.03         $(example 14)
```

where:

- `G#` ..... Actual conductor number (integer)
- `NA,NB` .... Actual number of the pair of nodes to which the conductor is connected (integer or smn.integer)
- `APA,APB` .. References to arrays of T vs. G points of the form: An or smn.An, where n is the array number
- `ACA,ACB` .. References to arrays of polynomial coefficients, of the form: An or smn.An
- `FA,FB` .... Multiplying factors input as floating point data values or as user constant references of the form: XKm, Km, smn.XKm, or smn.Km, where m is the user constant number

Using interpolation (or polynomial evaluation) as the calculation scheme, the first conductance, GA, is computed by using the temperature of node NA as the independent variable for operating on array APA (or ACA) with the result multiplied by FA. In similar fashion, the temperature of node NB, array APB (or ACB) and factor FB are used to compute the second conductance, GB. These two conductances are then combined to form the total conductance, GT, which is entered in memory. The combination algorithm depends on the type of the conductor, as follows:

Linear (series conduction): $GT = \left(GA^{-1} + GB^{-1} \right)^{-1}$

Radiation:* $GT = GA \cdot GB$

If one of the two materials does not have a temperature varying conductance, this material may be identified by replacing its corresponding array reference with a floating point data value. The constant conductance of this material will then be the product of this data value or constant and its associated multiplying factor. This conductance is combined with the variable conductance to form GT, as described above. For completeness, all possible alternate formats are listed, as follows:

---

* This option should be used with caution to avoid the calculation of an effective radiation conductance of

$$GT = \sigma^2 \mathcal{F}_1 \mathcal{F}_2 A_1 A_2$$

```
DIV G#,NA,NB,SUB,FA,APB,FB        $(GA = SUB*FA)
DPV G#,NA,NB,SUB,FA,ACB,FB        $(GA = SUB*FA)
DIV G#,NA,NB,APA,FA,SUB,FB        $(GB = SUB*FB)
DPV G#,NA,NB,ACA,FA,SUB,FB        $(GB = SUB*FB)
DPV 80,90,91,4.0,3.0,A6,K14       $(example 15)
```

where SUB is a floating point data value.

In example 14, the temperature of node 20 will be used with array 8 and constant 9 to compute GA, and the temperature of node 30 will be used with array 14 and the value 0.03 to compute GB. The total conductance, GT, will be entered for conductor number 10. In example 15, the value of GA will be computed as 12.0 (= 4.0*3.0), and the value of GB will be determined from T(91), A6, and K14.

DIM and DPM Options — These two options combine the features of the DIV and DPV options with those of the GEN option. They are used to generate and input a group of conductors all made of the same two materials having different temperature varying conductances. The DIM option causes the conductances for the two materials to be evaluated by interpolating on the referenced arrays, and the DPM option computes the conductances from polynomial coefficients supplied in the arrays. As for the other variables conductance options, multiplying factors are applied to the results of the operations on the arrays. Eleven data values are required to define each group of conductors, as follows:

```
C (basic format)
     DIM G#,#G,IG,NA,INA,NB,INB,APA,FA,APB,FB
C (basic format)
     DPM G#,#G,IG,NA,INA,NB,INB,ACA,FA,ACB,FB
C (example 16)
     DIM 10,3,2,20,1,30,-1,A12,K3,A13,K4
```

where:

G#	Actual conductor number of the first conductor to be generated (integer)
#G	Total number of conductors to be generated (integer)
IG	Increment to be added to the current conductor number to form the conductor number of the next conductor (non-zero integer)
NA,NB	Actual node numbers of the pair of nodes to which the first conductor is connected (integer or smn.integer)
INA,INB ..	Increments to be added to NA and NB, respectively, to form the NA, NB pair associated with the next conductor (integer)
APA,APB ..	References to arrays of T vs. G points, of the form: An or smn.An, where n is the number of the array being referenced.
ACA,ACB ..	References to arrays of polynomial coefficients, of the form: An or smn.An
FA,FB	Multiplying factors input as floating point data values, or as user constant references of the form: XKm, Km, smn.XKm, or smn.Km, where m is the actual constant number of a floating point value

Negative signs preceding G#, NA, or NB serve as flags to establish the type of the conductors, in accordance with the usual conventions. Negative signs preceding IG, INA, or INB, however, indicate that the data value is, indeed, a negative arithmetic quantity. Since the conductance of each conductor is dependent on the specific nodes to which it is connected, the IG value may not be zero (i.e., multiply-defined conductors are not allowed).

Since the DIM and DPM options are equivalent, in effect, to a series of conductors defined using the DIV or DPV options, the rules for evaluating the conductance of each conductor generated under the former pair of options are the same as those described in the section covering the latter pair of options.

For either option, one of the two materials may be defined as having a constant conductance by replacing the array reference for the material with a floating point data value. This feature makes the following alternate formats acceptable:

```
DIM G#,#G,IG,NA,INA,NB,INB,SUB,FA,APB,FB
DIM G#,#G,IG,NA,INA,NB,INB,APA,FA,SUB,FB
DPM G#,#G,IG,NA,INA,NB,INB,SUB,FA,ACB,FB
DPM G#,#G,IG,NA,INA,NB,INB,ACA,FA,SUB,FB
```

where SUB is a floating point data value.

The conductors defined in example 16 are equivalent to those which are defined by the following:

```
DIV 10,20,30,A12,K3,A13,K4
DIV 12,21,29,A12,K3,A13,K4
DIV 14,22,28,A12,K3,A13,K4
```

BIV Option — The BIV option enables the user to define a conductor having a conductance which is a function of both time and temperature. The card format for such a conductor is similar to that for the SIV option and requires five data values for each conductor, as follows:

```
BIV G#,NA,NB,APB,F                $(basic format)
BIV 19,8,11,A43, K28              $(example 17)
```

where:

G# ..... Actual conductor number (integer)

NA,NB .... Actual node numbers of the pair of nodes to which this conductor is connected (integer)

APB ..... Reference to a bivariate array (of temperature and time vs. conductance points) of the form: An or smn.An, where n is the array number

F ..... Multiplying factor input as a floating point data value or a user constant reference of the form XKm, Km, smn.XKm, or smn.Km, where m is the constant number of a floating point user constant

The temperature value used is the average value of the temperatures of the two nodes (NA, NB) to which the conductor is connected. A negative value of F is not permitted in this option. The time value used is the mean time for the computational interval (control constant TIMEM).

Section 3.7.2 describes the structure of a bivariate array. In this application, the X independent variable should be temperature, the Y independent variable should be time, and the Z dependent variable should be conductance.

Example 17 defines conductor 19, connected between nodes 8 and 11. Array 43 has a bivariate structure, and constant 28 contains a floating point value.

PIV and PIM Options – The PIV option enables the user to define a conductor having a conductance which is a function of both an arbitrary time dependent variable and temperature. The PIM option combines the features of the PIV option with those of the GEN option.

The arbitrary variable is first interpolated from the AT array as a function of time. This interpolated variable along with temperature are used as the independent variables entering into the bivariate array AAT to interpolate the conductance.

A common usage of these functions is interpolating the conductance of an insulation material where the conductance is a function of temperature and pressure. The pressure versus time history is entered into array AT and the conductance as a function of temperature and pressure is entered into array AAT.

To specify the PIV or PIM options the following data are entered into the conductor data block:

```
PIV G#,NA,NB,AT,FT,AAT,FAT
PIV G#,NA,NB,SUB,FT,AAT,FAT
PIV G#,NA,NB,AT,FT,SUB,FAT
PIM G#,#G,IG,NA,INA,NB,INB,AT,FT,AAT,FAT
PIM G#,#G,IG,NA,INA,NB,INB,SUB,FT,AAT,FAT
PIM G#,#G,IG,NA,INA,NB,INB,AT,FT,SUB,FAT
PIM 1,3,1,1,1,2,1,A6,K4,A20        $(example 18)
PIV 4,2,5,A6,0.870,A20,K2          $(example 19)
```

where:

G# ..... Actual conductor number (non-zero integer)

NA,NB .... Actual node numbers of the nodes to which the conductor is connected (integer or smn.integer - see Note 3)

#G ..... Total number of conductors to be generated (non-zero integer)

IG ..... Increment to the conductor number (integer)

INA,INB .. Increments to the node numbers, NA and NB, respectively (integer)

AT ..... Reference to doublet array of a variable (A) vs. time (t). A = f(t)

AAT ..... Reference to bivariate array of conductance (G) versus a temperature variable (T) and an arbitrary variable (A). G = f(T,A). The temperature used in calculation is the average of NA and NB. [Note 4].

FT,FAT ... Multiplication factors (floating point data values, or reference to user constants such as Km or smn.XKm.)

SUB ..... Floating point data value (literal) used as a substitute for an array reference. A user constant *cannot* be used.

Note 1: In both the PIM and PIV options when: SUB,FT,AAT,FAT is used, the arbitrary variable (A) is evaluated as A = SUB*FT. When: AT,FT,SUB,FAT is used, the arbitrary variable (A) is evaluated by interpolation. A = F(t); the conductance (G) is evaluated as G = A*FT*SUB*FAT.

Note 2: Conductor type is specified by the sign of the G# as follows:
LINEAR = positive conductor number
RADIATION = negative conductor number

Note 3: The negative signs on node numbers are considered flags and are ignored in the arithmetic sense. Preceding NA or NB with a minus sign will cause the conductor to be treated as a one-way conductor. The "upstream" node should be identified with the minus sign. The minus sign appears in the smn format as smn.-integer.

Note 4: Consistent with the bivariate array input format described in Section 3.7.2 the arbitrary variable (A) corresponds to the Y values and the temperature (T) corresponds to the X values. The units of the *arbitrary variable entered into the bivariate array* must be equal to the product of the arbitrary variable and the multiplication factor (FT) for the time dependent doublet array.

The following examples demonstrate the use of the PIV and PIM options. With reference to the Figure 3–8, it is desired to input the conductors as a function of pressure and temperature.

The following conditions are assumed:

1) A doublet array of pressure (Newtons/cm^2 versus time (sec) has been entered into the ARRAY DATA block as array no. 6.

2) A bivariate array of conductance (BTU/hr-°F) versus pressure (psi) and temperature (°F) has been entered into the ARRAY DATA block as array no. 20.

3) The constant 1.45 which changes Newtons/cm^2 to PSI has been entered into a USER DATA block as K4.

Consistent with the above conditions, conductors 1, 2 and 3 can be specified using the PIM option, as shown in Example 18.

The following conditions are assumed for conductor no. 4:

1) Orthotropic effects reduce the conductance to 60 percent of the table value. Therefore, 0.6 has been entered into the USER DATA block as K2.

2) The pressure is determined to be 60 percent of the table value. Therefore, the pressure multiplication factors will be 0.6 * 1.45 = 0.870 (the factor 1.45 changes Newtons/cm2 to psi).

Consistent with the above assumptions, conductor no. 4 can be specified using the PIV option, as shown in Example 19.

TVS and PER Options — These options allow the user to define time-variable conductors. The time dependency is determined by interpolating on two singlet time and conductance arrays which may appear in either ARRAY DATA or TARRAY DATA blocks. There is a restriction that both arrays must appear in the same block.

If a problem involves time-dependent radiation conductors, (e.g., an articulating spacecraft in orbit) a huge volume of array data may be involved. This is the reason for restricting this option to singlet arrays, which require a minimum number of separate data values. Another memory conservation device is the TARRAY DATA block. This approach stores the arrays on a disk file and only brings the data necessary for current interpolation into memory as required.

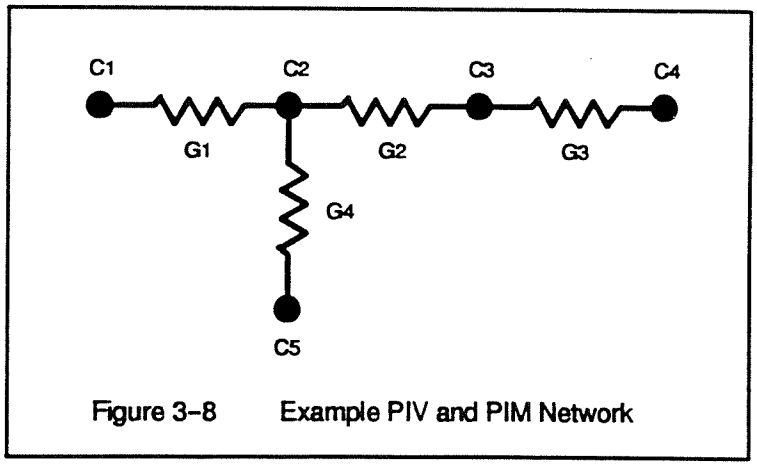


Figure 3–8 Example PIV and PIM Network

The TARRAY DATA block is the most memory-conserving, but imposes a processing time penalty because of the data transfer operations from files to core memory. The user should opt for the ARRAY DATA block unless the memory limit is exceeded. Switching from one to the other is a simple matter of changing the header record(s). Probably the most efficient mode, up to a point, is to forego the TVS and PER options, place the appropriate subroutine calls in VARIABLES 0 and use the ARRAY DATA block. This is practical if the subroutine calls can be provided by another computer program, (e.g., TRASYS) but when one or more thousands of subroutine calls appear in VARIABLES 0, Fortran compilation time will definitely be nontrivial. Some compilers will fail on such large routines. The point where this approach compares poorly with TVS & PER must be determined by the user for his particular computer system.

The ARRAY or TARRAY DATA formats required for TVS & PER are:

$$nt = t_1, t_2, t_3 - - - t_n$$
$$ng = g_1, g_2, g_3 - - - g_n$$

where:

- nt ..... time array number
- t_i ..... time values, strictly increasing with i
- ng ..... conductance array number
- g_i ..... conductance values (dependent variable) (t_i, g_i are all floating point data values)

The CONDUCTOR DATA formats are as follows:

```
    TVS G#,NA,NB,At,Ag,F
C (example 20):
    TVS 16,120,DOOR.18,A1,A1412,K12
    PER G#,NA,NB,At,Ag,F,D,P
C (example 21):
    PER 23,131,62,POD.A1,POD.A113,1.0,0.0,1.522
```

where:

G# ..... Conductor number, integer

NA,NB .... Actual node numbers to which the conductor is connected (integer or smn.integer)

At ..... Reference to singlet time array of the form An or smn.An where n is the actual array number

Ag ..... Reference to singlet conductance array of the form An or smn.An where n is the actual array number

F ..... Conductance multiplying factor, floating point number or user constant reference, form XKm, Km, smn.XKm, or smn.Km, where m is the actual constant number

D ..... Phase displacement as a decimal fraction of the total period, floating point value between 0.0 and 1.0. A user constant reference is not allowed

P ..... Period-specified in time units (floating point data value or reference to a user constant). P is usually the last time point in the At array, but not restricted to this value. If P is less than the last At array value, any values between P and the last array value are not used. If P is greater than the last array time value, the G value used for the time period between the last array time value and P is constant and equal to the G value for the last array time value

Example 20 defines a time-variable conductor no. 16 between node 120 and node 18 of submodel DOOR using time array 1 and conductance array 1412. The conductance value is multiplied by user constant 12.

Example 21 defines a time-variable conductor no. 23 between nodes 131 and 62. The time and conductance arrays are 1 and 113 of submodel POD, respectively. The period is 1.522 hours with no displacement offset.

3.10.2 Conductor Data Updating

The time and temperature dependent conductor data inputs defined in the CONDUCTOR DATA blocks are actually applied (that is, calculated and stored in the conductor locations) at the end of VARIABLES 0 and VARIABLES 1 for time and temperature-dependent operations, respectively. The BIV option (time and temperature dependent) is done in VARIABLES 1 prior to each temperature iteration.

These updates are done at the end of the VARIABLES 0 and VARIABLES 1 subroutines using calls to program subroutines GVTIME and GVTEMP, respectively. The preprocessor normally inserts these calls subsequent to any user-supplied logic in the VARIABLES 0 or VARIABLES 1 blocks. This gives the user an opportunity to update any user constants or arrays that may affect the conductor data calculations before they are performed. If user intervention is required after the GVTIME/GVTEMP calls, the user may supply a CALL GVTIME('smn') statement anywhere in the VARIABLES 0 input for submodel "smn" and it will not be overridden by a preprocessor-inserted call. This is also true for GVTEMP in VARIABLES 1.

All conductor value referencing in the logic blocks is done using the forms: Gn or smn.Gn where smn is a submodel name and n is the actual conductor number. As usual, the "smn" prefix is required only in the OPERATIONS DATA block and for cross-references to a node outside the submodel stated in the header card currently in effect. With this referencing capability, the user may use, modify, or print G-values in the logic blocks.

3.11 FLOW DATA

3.11.1 Basic Concepts

This section describes the input methods and formats for FLOW DATA, which is the main input block for fluid submodel descriptions. Fluid submodels are analyzed by FLUINT (FLUid INTegrator), which solves fluid network in parallel with the thermal network solutions provided by SINDA. Readers unfamiliar with FLUINT should first read Section 4.7: Fluid System Modeling, which explains modeling options and program operations. Such information will not be repeated here except where it is needed to clarify input procedures. The reader should note that the input rules for fluid data differ significantly from those for thermal data. The FLOW DATA input rules were selected to reduce the amount of work ultimately required by the user, although an initial amount of effort is needed to learn these rules.

A fluid network or *submodel* is composed of three types of elements: *lumps, paths,* and *ties.* Lumps are point locations where mass and energy are conserved. They exchange mass and energy with each other via paths. Paths are governed by momentum considerations and/or other relationships between flowrate and pressure gradient, depending on type. Ties are used to connect fluid submodels to thermal submodels for heat transfer; they are analogous to intermodel thermal conductors.

Lumps are subdivided into three types: *tanks, junctions,* and *plena.* Tanks are finite control volumes that may exchange energy with the environment and may grow or shrink. Because they are governed by differential equations for mass and energy in time, tanks cannot change instantaneously. Junctions can be conceptualized as zero volume tanks that may also exchange energy with the environment. Junctions react instantaneously because they are governed by algebraic (time-independent constraint) equations for mass and energy. They may be used to model tees, small tanks, or points where flow information is needed. Plena (plural of "plenum") can be conceptualized as tanks with infinite volume. They represent boundary conditions for modeling open systems. Each lump has a single characteristic thermodynamic state, and therefore inherently presumes perfect mixing. To simulate lack of thermal equilibrium between phases when liquid and vapor are both present, pairs of tanks may be input and nonequilibrium simulation routines may be invoked.

Paths are subdivided into two types: *tubes* and *connectors.* Tubes are paths with significant line inertia; they are governed by a differential equation for momentum and cannot change flowrate instantaneously. Connectors are paths with insignificant inertia (flowrates may change instantaneously), and are governed by an algebraic relationship between flow rate, pressure differential, etc. Connectors may be used to simulate pumps, valves, loss factors, capillary passages, leaks, and short lines. Each path has only one characteristic flowrate and velocity, and are therefore behave homogeneously when two–phase flow is flowing through them. To simulate slip flow, assigning one characteristic flowrate and velocity to each phase, paths may be input in pairs called *twins.*

Ties are subdivided into six types: *HTN, HTNC, HTNS, HTU, HTUS,* and *HTM.* The first three types represent convective heat transfer options. The fourth and fifth types (HTU and HTUS) represent user-defined heat transfer processes. The last type is generated by multi-element program models, and has different meanings in each model. (See also Section 4.)

Comparisons to thermal models and elements may help explain the distinctions presented above. Lumps are analogous to thermal nodes, and paths are analogous to conductors. Furthermore, tanks are analogous to diffusion nodes, as are junctions to arithmetic nodes and plena to boundary nodes. The conceptual analogies between path types and conductor types are more strained and should be avoided. Note that thermal submodels may *not* include lumps or paths, and that fluid models may *not* include nodes or conductors. However, lumps and nodes can be linked together with ties, which are very similar to SINDA conductors. Lumps represent the fluid inside a container, nodes (if desired) represent the container itself, and ties represent the heat transfer process linking the two elements.

A fluid submodel represents a fluid system that contains a single fluid substance and that does not exchange fluid with any other submodel. Each fluid submodel is described within an input section that begins with a HEADER FLOW DATA card. Following this header card, the user specifies the submodel name and desired working fluid, the fluid network, initial conditions, and calculation options. There are four basic types of *subblocks* within this section that may appear *in any order* to facilitate reading from a schematic. These subblocks begin with reserved letters in column 1 (and sometimes extends into column 2), and terminate when another subblock or header record is encountered. The four subblock types are:

LU ..... Lump input subblock
PA ..... Path input subblock
M ..... Macro or Model subblock (for generation of elements or component models)
T ..... Tie subblock, used to specify energy exchange with thermal nodes

These subblock identifiers (LU, PA, M, and T) must appear in column 1. The remainder of the inputs may be given in any column and may extend to several lines. The general format of any fluid subblock is:

```
id option, required input #1, required input #2, ...
     [,keyword=value] [,keyword=value]
     [,keyword=value]  ...
```

where:

`id` ..... subblock identifier (LU, PA, M or T)
`option` ... name of the subblock type (such as TANK, TUBE, etc.)

Following the *option* specification is a list of required inputs (i.e., inputs with no definable defaults) that are separated by commas. Following the required inputs, the user may specify any options that are specific to this *id* and *option*. These optional inputs are identified by *keywords*, and may occur in any order (with one exception noted in 3.11.2.3). The user may specify some, all, or none of these inputs. In the format descriptions given in the next section, different values will be expected depending on the keyword used:

Value	Expected Input
R	Real (floating point) input expected
I	Integer input expected
C	Character (alphanumeric) input expected, without quotes
A	User array ID ([smn.] 21, 3, etc.) expected.

Fortran-style arithmetic (+, -, *, and /) is allowed in the input fields. *Note that such operations are evaluated from left to right with no precedence given to * or / operations.* Parentheses, functions, and references to user constants are not allowed. Operation on two integers produces an integer result; otherwise all numbers are treated as floating point. Unlike thermal data input blocks, *named or numbered user constants cannot be referenced in FLOW DATA blocks.*

A backslash is not necessary to continue a subblock. Subblocks may extend over several records, and are terminated when a new subblock or header is encountered. Whole lines may be specified as comments by beginning the line with a C in column 1. These lines are ignored by the program. A dollar sign ($) similarly causes all subsequent information on a line to be ignored, allowing comments at the end of the line. Comments do not terminate the current subblock.

In order to reduce the amount of input required, the user has the option of using *DEF* ("define" or "default") subblocks. With these subblocks, the user may preset values for information that would otherwise need to be repeated for each element. All input for DEF subblocks is optional* and therefore keyword-driven. There are two types of DEF subblocks: one for lumps (LU DEF) and one for paths (PA DEF). Any values defined in DEF subblocks become default values for all subsequent input in the current submodel. The user may use DEF subblocks as often as needed.

T subblocks are used to link together nodes and lumps and to select the methodology for heat transfer between fluid and thermal submodels. Three options are currently available: HTN, HTNC, HTNS, HTU, and HTUS. A sixth type, HTM, is an internal designation that is generated by certain macrocommands.

M subblocks are used to generate many elements in subnetworks, similar to the GEN options in thermal input sections. Like GEN options, the generated elements may be independent (such as a string of lumps and paths in a subnetwork). Unlike GEN options, the generated elements may instead be logically linked to form *component models* (such as ducts and capillary pumps).

Special note should be made of the methods used to specify initial thermodynamic states for lumps. There are three inputs allowed: temperature, vapor quality**, and pressure (TL, XL, and PL, respectively). In order to allow convenient input of two-phase states (in which temperature and pressure are not independent properties), a logical hierarchy exists:

1) TL is used

2) XL is always used

3) PL is overridden if it conflicts thermodynamically with TL and XL.

Table 3–9 explains how to specify states given the potential conflict between XL and PL. Note that PL is only needed for subcooled liquid or superheated vapor states. A very low PL (say 0.0) when XL=0.0 signals that saturated (rather than subcooled) liquid is desired, whereas a very high PL (say 1.0E30) when XL=1.0 signals that saturated (rather than superheated) vapor is desired.

---

* Note that some keyword-driven "options" are only optional *if they have been defined previously in the current submodel using a DEF subblock.* An error will result if no default value has been defined and no input is given in the current subblock.

** Quality is defined as the ratio of vapor to total mass.

Table 3–9 Lump State Initialization (without "PL!")

Desired State	Input Quality	Input Pressure
Subcooled Liquid	XL = 0.0	PL > P_{sat}(TL) *
Saturated Liquid	XL = 0.0	PL ≤ P_{sat}(TL) **
Two-phase	0.0 < XL < 1.0	anything **
Saturated Vapor	XL = 1.0	PL ≥ P_{sat}(TL) **
Superheated Vapor	XL = 1.0	PL < P_{sat}(TL)

* P_{sat}(TL) is the saturation pressure corresponding to TL.

** PL will be replaced with P_{sat}(TL) in these cases.

Initial pressure is thus ordinarily considered of secondary importance. If this is not true, then an exclamation point ("!") after the PL signals that the pressure is of primary importance, and that temperature is to be overridden if necessary:

1) PL! is always used

2) XL is always used

3) TL is overridden if it conflicts thermodynamically with PL! and XL.

If PL! is used, the value of TL can be purposely over- or undersized to force its override in by a methodology analogous to Table 3–9. PL! may be specified in LU DEF blocks, in which case that value applies to all following lumps in which *neither* PL nor PL! is specified: a local subblock value of PL is assumed to have priority over a default value even though that local value may be overridden if inconsistent. The exclamation point can only occur in FLOW DATA for initial conditions, and has no meaning in logic blocks.

XL is always obeyed unless the working fluid is inherently single–phase (refer to FPROP DATA in Section 3.13) is used, in which case XL is ignored as redundant and PL is applied as input.

The following FLUINT variables must be input at least once (i.e., they have no default value until assigned one in a DEF subblock): PL, TL, VOL, FR, TLEN, and DH. When these values are missing from the examples that follow, they are assumed to have been set in a prior DEF subblock.

A sample submodel input deck is provided in Section 3.11.3 to help clarify the *input format* concepts introduced here. Refer to Appendix F (in a separate volume) for detailed descriptions of more sample problems. Section 6.9 contains descriptions of FLUINT utility routines. Refer to

Appendix C and Section 3.13 for acceptable fluid identifiers, and refer to Appendix D for FLUINT unit conventions. Other information relating to correlations and numerical methods may be found in Appendices A, B, and E.

3.11.2 Fluid Submodel Input Formats and Definitions

3.11.2.1 Submodel-Level Inputs (HEADER Block)

Unlike thermal submodel inputs, almost all inputs for fluid submodels (network element descriptions and source terms) are input under *one* HEADER block. Within this block, subblocks describing network elements may occur in any order. This style facilitates walking through fluid system schematics. The format for this HEADER block is:

```
HEADER FLOW DATA,smn[,FID=I][,NWORK=I]
```

where:

`smn`	fluid submodel name (must be unique, up to 8 characters)
`FID`	fluid identification number (ASHRAE number or user fluid ID number—see also Appendix C and Section 3.13)
`NWORK`	size of allocated workspace (memory) for sparse matrix inversion. ***This is rarely used***; do not use this option unless (1) a processor error message has requested it and specified a value to use, or (2) the YSMPWK routine has been correctly used to analyze memory requirements, and the risk of an aborted run is deemed to be worth the reduced allocation of memory.

defaults:

`FID`	718 (water, full two-phase library description)
`NWORK`	calculated by program

Examples:

```
HEADER FLOW DATA, LOOP, FID=11
HEADER FLOW DATA, ATCS
```

3.11.2.2 Fluid Lump Specification (LU Subblock)

This subblock is used to input single lumps. There are four options available:

TANK ..... Tank (control volume)
PLEN ..... Plenum (boundary or infinitely large volume)
JUNC ..... Junction (point or infinitely small volume)
DEF ..... Used to define defaults for lumps

TANK Subblock Format:

```
LU TANK,id[,VOL=R][,TL=R][,XL=R][,PL(!)=R][,VDOT=R]
     [,QDOT=R][,COMP=R][,CX=R][,CY=R][,CZ=R]
```

where:

id ..... tank identification number
VOL ..... initial tank volume
TL ..... initial temperature
XL ..... initial quality
PL ..... initial pressure, PL! means PL value given priority over TL
VDOT ..... initial time rate of change of tank volume
QDOT ..... initial rate of heat input
COMP ..... initial tank wall compliance factor
CX ..... coordinate location on the X axis
CY ..... coordinate location on the Y axis
CZ ..... coordinate location on the Z axis

defaults (if not previously defined in an LU DEF subblock):

XL ..... zero
VDOT ..... zero (volume constant over time)
QDOT ..... zero (Note: QDOT will be overwritten if this lump has any ties, but QDOTs are not zeroed as are nodal Qs)
COMP ..... zero (inflexible wall)
CX ..... zero
CY ..... zero
CZ ..... zero

Guidance: Pressure spikes and notches may occur if tanks are hard-filled with liquid and inflexible (COMP=0.0). A small compliance factor (see Section 4.7) can avoid such problems, and should be considered mandatory for liquid tanks next to valves and capillary devices.

Restrictions: A tank must be linked via paths to at least one but not more than 25 other lumps. *If all tanks in a network are or can become hard-filled with liquid and are inflexible (COMP=0.0), at least one plenum must be present or a fatal error will result.* In this sense, a "network" is any isolated or valved-off part of the submodel.

Examples:

```
LU TANK, 10, VOL=3.0*4.0*5.0,TL=300.0
LU TANK, 20, QDOT=50.0,VOL=7.0, COMP=1.0E-3
```

PLEN Subblock Format:

```
LU PLEN,id[,TL=R][,XL=R][,PL(!)=R]
     [,CX=R][,CY=R][,CZ=R]
```

where:

id ..... plenum identification number
TL ..... plenum temperature
XL ..... plenum quality
PL ..... plenum pressure, PL! means PL value given priority over TL
CX ..... coordinate location on the X axis
CY ..... coordinate location on the Y axis
CZ ..... coordinate location on the Z axis

defaults (if not previously defined in an LU DEF subblock):

XL ..... zero
CX ..... zero
CY ..... zero
CZ ..... zero

Restrictions: A plenum may be linked via paths to any number of lumps. A warning will be produced if the plenum is isolated (i.e., unconnected).

Examples:

```
LU PLEN,77,PL!=14.7
LU PLEN,18,TL=200.0,XL=0.5,CZ=-10.0
```

JUNC Subblock Format:

```
LU JUNC,id[,TL=R][,XL=R][,PL(!)=R][,QDOT=R]
     [,CX=R][,CY=R][,CZ=R]
```

where:

TL ..... initial temperature
XL ..... initial quality
PL ..... initial pressure, PL! means PL value given priority over TL
QDOT ..... initial rate of heat input
CX ..... coordinate location on the X axis
CY ..... coordinate location on the Y axis
CZ ..... coordinate location on the Z axis

defaults (if not previously defined in an LU DEF subblock):

XL ..... zero
QDOT ..... zero (Note: QDOT will be overwritten later if this lump has any ties or becomes a heater junction via a call to HTRLMP, but QDOTs are not zeroed as are nodal Qs)
CX ..... zero
CY ..... zero
CZ ..... zero

Guidance: Junctions are less expensive to analyze than are tanks, and should be used instead of tanks whenever possible. Such substitutions are not *always* preferred.

Restrictions: A junction must be linked via paths to at least one but not more than 25 other lumps. No submodel can be built entirely from junctions. Also, fatal errors will result if a closed loop exists in the network that is composed solely of junctions and QDOT values for the loop are not zero or do not exactly balance (except in FASTIC).

Examples:

```
LU JUNC,3010,QDOT=410.0,XL=0.1
LU JUNC,14,  PL=1.0E7,  CX=1.0, CZ=2.0, CY=-3.0
```

LU DEF Subblock Format:

```
LU DEF[,VOL=R][,TL=R][,TLADD=R][,TLFACT=R][,XL=R]
     [,PL(!)=R][,PLADD=R][,PLFACT=R][,VDOT=R]
     [,QDOT=R][,COMP=R][,CX=R][,CY=R][,CZ=R]
```

where:

VOL ..... initial volume (for tanks)
TL ..... initial temperature
XL ..... initial quality
TLADD .... temperature offset
TLFACT ... temperature multiplier
PL ..... initial pressure, PL! means PL value given priority over TL
PLADD .... pressure offset
PLFACT ... pressure multiplier
VDOT ..... initial time rate of change of volume (for tanks)
QDOT ..... initial rate of heat input (for tanks and junctions)
COMP ..... initial tank wall compliance factor
CX ..... coordinate location on the X axis
CY ..... coordinate location on the Y axis
CZ ..... coordinate location on the Z axis

defaults (if not previously defined in an LU DEF subblock):

TLADD .... zero
TLFACT ... one
XL ..... zero
PLADD .... zero
PLFACT ... one
VDOT ..... zero
QDOT ..... zero (Note: QDOT will be overwritten later if this lump is tied, but QDOTs are not zeroed as are nodal Qs)
CX ..... zero
CY ..... zero
CZ ..... zero

Guidance: An LU DEF subblock may be used to define default values for the lumps following the subblock. Any number of LU DEF subblocks may be used in any order to set or change default values as needed. These defaults operate only within the current submodel, and stay in effect until specifically overridden by another LU DEF sub-

block. Inputs given in any other LU subblock do not affect these defaults. Offsets (TLADD, PLADD) and multipliers (TLFACT, PLFACT) can be used to change units or reference temperatures and pressures. These factors apply *only during input* to subsequent TLs and PLs in the following manner:

Actual Value = (input value + offset)*multiplier

Guidance: The use of PL without an exclamation mark will erase any previous LU DEF subblock that did use PL!, meaning that PL is once again second in priority to TL. Because explicit LU subblock inputs *always* override any LU DEF subblock inputs, the explicit use of PL (without the "!") in a subsequent lump subblock will override the presence of PL! used in an earlier LU DEF subblock.

Examples:

```
LU DEF, TL=300.0, PL=0.0, TLFACT=1.8, PLADD=14.7
LU DEF, XL=0.0, QDOT=100.0, CX=10.0
```

3.11.2.3 Flow Path Specification (PA Subblock)

This subblock is used to input single paths, or single pairs of *twinned paths*. There are three options available:

TUBE ..... Tube, a time-dependent (inertial) path
CONN..... Connector, a time-independent (instantaneous) path
DEF ..... Used to define defaults for paths

Connectors are further subdivided by device type, as detailed below.

Paths may be assigned duplication factors, DUPI and DUPJ, which are both unity by default. These values may be used to exploit symmetries such as parallel passages, or to shut off paths (by setting them to zero). DUPI is the factor applied to the defined upstream end of the path, and DUPJ is the factor applied to the defined downstream end. The upstream lump "sees" DUPI such paths, and the downstream lump "sees" DUPJ such paths. Thus, the total flowrate exiting the upstream lump via that path is DUPI•FR, while the total flowrate entering the downstream lump is DUPJ•FR.

By default, paths are single, homogeneous entities. Alteratively, they may be input in permanently matched pairs or twins. The first or *primary* twin normally transports liquid, while the second or *secondary* twin normally transports vapor.

For tubes and STUBE connectors, slip flow modeling is automatically invoked when paths are twinned. This feature is intended to be used in conjunction with IPDC=6, where IPDC is the two–phase frictional pressure drop correlation identifier defined later. When only single phase fluid is present in tubes or STUBE connectors, the secondary twin will disappear and only the primary path will exist. It will behave as if no twin existed. When two–phase flow reappears, the primary path will revert to carrying liquid and the secondary path will reappear, carrying vapor. Refer to Section 4.7.

Any other connector device may also be twinned. Two identical paths are generated, the first with STAT initialized to DLS and the second with STAT initialized to DVS. However, unlike tubes or STUBE connectors, no other modeling logic is invoked for other twinned devices. (An minor exception to this statement is the use of twinned NULL connectors in the nonequilibrium routines.)

TUBE Subblock Format:

```
PA TUBE,id,l1,l2[,FR=R][,TWIN=I][,STAT=C]
      [,TLEN=R][,DH=R][,AF=R][,AFTH=R][,FK=R]
      [,WRF=R][,IPDC=I][,UPF=R]
      [,HC=R][,FC=R][,FPOW=R][,AC=R][,AM=R][,FG=R][,FD=R]
      [,DUPI=R][,DUPJ=R][,DUP=R]
```

where:

`id` ..... tube identifier

`l1,l2` .... upstream and downstream lump identifiers

`FR` ..... initial mass flow rate

`TWIN` ..... unique identifier of secondary twin tube to be generated (for slip flow modeling), where id is the primary twin

`STAT` ..... phase suction status flag:

- NORM: normal (extracts both phases from upstream lump)
- LS: Liquid suction from l1 only
- VS: Vapor suction from l1 only
- DLS: Liquid suction from l1 *or* l2
- DVS: Vapor suction from l1 *or* l2
- RLS: Liquid suction from l2 only
- RVS: Vapor suction from l2 only

`TLEN` ..... tube length

`DH` ..... hydraulic diameter (if twinned, applies to whole passage)

`AF` ..... flow area (if twinned, applies to whole passage)

`AFTH` ..... throat flow area for optional choking calculations

`FK` ..... K–factor due to associated entrance, exit, or bend losses

`WRF` ..... wall roughness fraction

`IPDC` ..... Two-phase frictional pressure drop correlation number:

- 0: User-input FC and FPOW even for single-phase flow
- 1: McAdam's Homogeneous
- 2: Lockhart-Martinelli
- 3: Baroczy
- 4: Friedel
- 5: Whalley's recommendations
- 6: Flow regime based two–phase friction

`UPF` ..... (if IPDC isn't 0) upstream fraction of properties to use in friction pressure drop calculations

`HC` ..... initial head coefficient (impressed pressure gradient)

`FC` ..... initial irrecoverable loss coefficient

`FPOW` ..... flowrate exponent in frictional pressure drop term: $FC \cdot FR \cdot |FR|^{FPOW}$

`AC` ..... initial recoverable loss coefficient

AM ..... initial added (virtual) mass coefficient, for twinned tubes
FG ..... initial phase generation coefficient, for twinned tubes
FD ..... initial interface friction coefficient, for twinned tubes
DUPI ..... Upstream duplication factor (number of this path that appear to l1)
DUPJ ..... Downstream duplication factor (number of this path that appear to l2)
DUP ..... Duplication factor (number of this path that appear to *both* l1 and l2; sets both DUPI and DUPJ in preprocessor only)

defaults (if not previously defined in a PA DEF subblock):

TWIN ..... no twin (single homogeneous path created)
STAT ..... NORM
AF ..... -1.0 (nonpositive value signals that AF should be calculated by assuming a circular cross section of diameter DH. Otherwise, DH and AF are independent parameters.)
AFTH ..... AF
FK ..... zero
WRF ..... zero
IPDC ..... 1 (McAdam's Homogeneous); 6 (flow regime mapping) if twinned
UPF ..... 0.5 (Average of up and downstream properties)
HC ..... zero
FC ..... zero (calculated automatically if IPDC isn't 0)
FPOW ..... 1.0 (calculated automatically if IPDC isn't 0)
AC ..... zero
AM ..... zero (calculated automatically if twin tubes are input)
FG ..... zero
FD ..... zero (calculated automatically if twin tubes are input and IPDC=6)
DUPI ..... 1.0 (or value of DUP)
DUPJ ..... 1.0 (or value of DUP)
DUP 1.0

Guidance: Order of l1,l2 indicates positive flow direction.

Guidance: While the parameters FC, FPOW, AC, AM, FG, and FD are available for direct input, they are normally calculated automatically by devices options and their presence can be ignored by casual users.

Guidance: If a frictional pressure drop model other than those available is desired, or if FC is to be zero (no frictional losses) or otherwise provided by user logic, set IPDC=0. If IPDC=0, then the input values of TLEN, WRF, UPF, and DH are ignored although TLEN and DH might still be used by HTN, HTNC, and HTNS ties. If IPDC=0, the user assumed control of both FC and FPOW.

Guidance: IPDC=6 should be used with twinned tubes, otherwise the interface friction term FD will remain at the unrealistic (but legal) limit of zero.

Guidance: If twinned, the initial flowrate is assumed to be the total flowrate. It will be apportioned to each twin on the basis of initial upstream quality.

Guidance: If twinned, all other initial values apply to both twins. If FC, FPOW, AC, AM, FG, or FD are being specified in FLOW DATA for twinned tubes, the user should consider modifying these values for each twin in OPERATIONS DATA prior to a solution routine call since these parameter will rarely be equal for both twins. In fact, the magnitude of AM, FG, and FD should normally differ by the ratio of densities, and FG should be of opposite sign for each phase.

Restriction: Phase suction options (STAT) are ignored in conjunction with twinned tubes (see also GOHOMO and GOSLIP support subroutines in Section 6.)

Restriction: If IPDC=0, all of FK, FC, and AC cannot be zero in FASTIC, and they should not all be zero indefinitely in other solution routines.

Examples:

```
PA TUBE,10,1,2, FR=0.03, FK=2.0, IPDC=5, UPF=1.0, DUP=500.0
PA TUBE,13,100,200, DH=1.0/12.0, AF=1.0, WRF=1.0E-4
PA TUBE,203,2,3,       TWIN = 1203    $ GENERATES TWO TUBES
                       IPDC = 6
```

CONN Subblock Format:

```
PA CONN,id,l1,l2[,FR=R][,TWIN=I][,STAT=C]
     [,GK=R][,HK=R][,EI=R][,EJ=R][,DK=R]
     [,DUPI=R][,DUPJ=R][,DUP=R]
     [,DEV=C,...,...]
```

where:

`id` ..... connector identifier

`l1,l2` .... upstream and downstream lump identifiers

`FR` ..... initial mass flow rate

`TWIN` ..... unique identifier of secondary twin connector to be generated (for slip flow modeling), where id is the primary twin

`STAT` ..... phase suction status flag:

- NORM: normal (extracts both phases from upstream lump)
- LS: Liquid suction from l1 only
- VS: Vapor suction from l1 only
- DLS: Liquid suction from l1 *or* l2
- DVS: Vapor suction from l1 *or* l2
- RLS: Liquid suction from l2 only
- RVS: Vapor suction from l2 only

`GK` ..... initial derivative of flow rate with respect to pressure drop

`HK` ..... initial flow rate offset

`EI` ..... initial derivative of flowrate with respect to defined upstream enthalpy

`EJ` ..... initial derivative of flowrate with respect to defined upstream enthalpy

`DK` ..... initial derivative of flowrate with respect to twin's flowrate

`DUPI` ..... Upstream duplication factor (number of this path that appear to l1)

`DUPJ` ..... Downstream duplication factor (number of this path that appear to l2)

`DUP` ..... Duplication factor (number of this path that appear to *both* l1 and l2; sets both DUPI and DUPJ)

`DEV` ..... routine name for device simulation. May require extra inputs depending on device type. *Any such device specific inputs must appear at the* **end** *of the subblock following the DEV=C in the CONN subblock.* Refer to Section 3.11.2.4 for formats.

defaults (if not previously defined in a PA DEF subblock):

TWIN ..... no twin (single homogeneous path created)
STAT ..... NORM
GK ..... (required if DEV=NULL, otherwise calculated automatically by device model)
HK ..... zero (calculated automatically if device model is chosen)
EI ..... zero (calculated automatically if twinned STUBEs are chosen)
EJ ..... zero (calculated automatically if twinned STUBEs are chosen)
DK ..... zero (calculated automatically if twinned STUBEs are chosen)
DUPI ..... 1.0 (or value of DUP)
DUPJ ..... 1.0 (or value of DUP)
DUP 1.0
DEV ..... NULL (Warning: This option is for experienced users only!)

Guidance: Order of l1,l2 indicates positive flow direction.

Guidance: NULL connectors are rarely used, and in fact should only be used by experienced users. Thus, while the parameters GK, HK, EI, EJ, and DK are available for direct input, they are normally calculated automatically by devices options and their presence can be ignored by casual users.

Guidance: If twinned, the initial flowrate is assumed to be the total flowrate. It will be apportioned to each twin on the basis of initial upstream quality.

Guidance: If twinned, all other initial values apply to both twins. If GK, HK, EI, EJ, or DK are being specified in FLOW DATA for twinned paths, the user should consider modifying these values for each twin in OPERATIONS DATA prior to a solution routine call since these parameter will rarely be equal for both twins.

Restriction: Phase suction options (STAT) are ignored in conjunction with twinned paths (see also GOHOMO and GOSLIP support subroutines in Section 6.)

Restrictions: Either a device model must be specified, or GK and HK must be provided and manipulated by the user logic. Nonpositive GK should be used cautiously.

Restrictions: Note that the DEV option provides for a variable subset of inputs (and therefore keywords). With the exception of the STUBE model, no defaults may be specified for these inputs in PA DEF subblocks.

Examples:

```
PA CONN,1,15,16,STAT=DLS, FR=17.0, DEV=MFRSET
PA CONN,777,101,102, GK=1.0E-3
PA CONN,12,443,444, TWIN = 1012, GK = 1.0
                          DEV     = NULL
```

PA DEF Subblock Format:

```
PA DEF[,FR=R][,FRFACT=R][,STAT=C]
      [,TLEN=R][,DH=R][,AF=R][,AFTH=R]
      [,WRF=R][,IPDC=I][,UPF=R]
      [,HC=R][,FC=R][,FPOW=R][,AC=R][,AM=R][,FG=R][,FD=R]
      [,GK=R][,HK=R][,EI=R][,EJ=R][,DK=R]
```

where:

FR ..... initial mass flowrate

FRFACT ... mass flowrate multiplier

STAT ..... phase suction status flag:

- NORM: normal (extracts both phases from upstream lump)
- LS: Liquid suction from l1 only
- VS: Vapor suction from l1 only
- DLS: Liquid suction from l1 *or* l2
- DVS: Vapor suction from l1 *or* l2
- RLS: Liquid suction from l2 only
- RVS: Vapor suction from l2 only

TLEN ..... tube length (tubes and STUBE connectors)

DH ..... hydraulic diameter (tubes and STUBE connectors; used by LOSS, LOSS2, CTLVLV, and CHKVLV connectors if AF is neither input nor defaulted)

AF ..... flow area (tubes and STUBE, LOSS, LOSS2, CTLVLV, CHKVLV and connectors)

AFTH ..... throat flow area for optional choking calculations (tubes and STUBE, LOSS, LOSS2, CTLVLV, and CHKVLV connectors)

WRF ..... wall roughness fraction (tubes and STUBE connectors)

IPDC ..... Two-phase frictional pressure drop correlation number:

- 0: User-input FC and FPOW even for single-phase flow
- 1: McAdam's Homogeneous
- 2: Lockhart-Martinelli
- 3: Baroczy
- 4: Friedel
- 5: Whalley's recommendations
- 6: Flow regime based two–phase friction

UPF ..... (if IPDC isn't 0) upstream fraction of properties to use in friction pressure drop calculations (tubes and STUBE connectors)

HC ..... initial head coefficient (tubes and STUBE connectors)

FC ..... initial irrecoverable loss coefficient (tubes and STUBE connectors)

FPOW ..... flowrate exponent in frictional pressure drop term: $FC \cdot FR \cdot |FR|^{FPOW}$ (tubes and STUBE connectors)

AC ..... initial recoverable loss coef. (or connector AC when DEV=STUBE)

AM ..... initial added (virtual) mass coefficient, for twinned tubes and STUBEs
FG ..... initial phase generation coefficient, for twinned tubes and STUBEs
FD ..... initial interface friction coefficient, for twinned tubes and STUBEs
GK ..... initial derivative of flow rate with respect to pressure drop
HK ..... initial flow rate offset
EI ..... initial derivative of flowrate with respect to defined upstream enthalpy
EJ ..... initial derivative of flowrate with respect to defined upstream enthalpy
DK ..... initial derivative of flowrate with respect to twin's flowrate

defaults (if not previously defined in a PA DEF subblock):

FRFACT ... one
STAT ..... NORM
AF ..... calculated using DH, assuming circular cross section. Note: negative AF signals a circular cross section, and AF will be overwritten. This allows users to override previous defaults for noncircular cross sections.
AFTH ..... AF
WRF ..... zero
HC ..... zero
FC ..... zero (calculated automatically if IPDC isn't 0)
FPOW ..... 1.0 (calculated automatically if IPDC isn't 0)
AC ..... zero (calculated automatically if paths are part of a duct macro)
IPDC ..... 1 (McAdam's Homogeneous); 6 (flow regime based pressure drop) if twinned tubes or STUBEs are chosen.
UPF ..... 0.5 (Average of up and downstream properties)
AM ..... zero (calculated automatically if twin tubes are input)
FG ..... zero (calculated automatically if paths are part of a duct macro)
FD ..... zero (calculated automatically if twin paths are input and IPDC=6)
GK ..... (required if DEV=NULL, otherwise calculated automatically)
HK ..... zero (calculated automatically if device model is chosen)
EI ..... zero (calculated automatically if twinned STUBEs are chosen)
EJ ..... zero (calculated automatically if twinned STUBEs are chosen)
DK ..... zero (calculated automatically if twinned STUBEs are chosen)

Guidance: A PA DEF subblock may be used to define default values for the paths following the subblock. Any number of PA DEF subblocks may be used in any order to set or change default values as needed. These defaults operate only within the current submodel, and stay in effect until specifically overridden by another PA DEF subblock. Inputs given in any other PA subblock do not affect these defaults.

Guidance: FRFACT is used to change units or reference status. This factor applies to all subsequent flowrates *during input only* as follows:

Actual FR = (input FR)*FRFACT

Guidance: While the parameters FC, FPOW, AC, AM, FG, FD, GK, HK, EI, EJ, and DK are available for direct input, they are normally calculated automatically by various options and their presence can be ignored by casual users.

Restriction Note the absence of duplication factors (DUPI and DUPJ). The defaults for these are always 1.0, which may not be reset because of the possibility for major errors that are difficult to detect.

Examples:

```
PA DEF,DH=0.01, TLEN=1.0, AF=1.0E-4
PA DEF,IPDC=1, UPF=0.0, FRFACT=0.4536/3600.0
```

3.11.2.4 DEV Model Specifications for PA CONN Subblocks

This section lists the formats used *in the second half of the PA CONN subblocks* after the DEV=C specification, where C is one of the following mnemonics:

NULL ..... No device (special path or user-controlled GK, HK, EI, EJ, DK)
STUBE ..... Short (instantaneous, time independent) tube
CAPIL ..... Capillary (surface tension dominated) passage
LOSS ..... Generic head loss component
LOSS2 ..... Two way head loss component
CHKVLV .... Check valve (shuts to prevent reverse flow)
CTLVLV ..... Control valve (shuts by command)
MFRSET .... Maintains a constant mass flow rate
VFRSET..... Maintains a constant volumetric flow rate
PUMP ..... Centrifugal pump, constant shaft speed
VPUMP ..... Centrifugal pump, variable shaft speed

Restrictions: *Any device-specific inputs must be located at the end of the CONN subblock following DEV=C*, although they may appear in any order after DEV=C. In other words, if an initial FR is given, it must appear before the DEV=C. Only one device specification is allowed per connector.

STUBE Format In CONN Subblocks:

```
... DEV=STUBE [,TLEN=R][,DH=R][,AF=R][,AFTH=R][,FK=R]
              [,WRF=R][,IPDC=I][,UPF=R]
              [,HC=R][,FC=R][,FPOW=R][,AC=R][,FG=R][,FD=R]
```

where:

TLEN ..... length
DH ..... hydraulic diameter
AF ..... flow area (if twinned, applies to whole passage)
AFTH ..... throat flow area for optional choking calculations
FK ..... K–factor due to associated entrance, exit, or bend losses
WRF ..... wall roughness fraction
IPDC ..... Two-phase frictional pressure drop correlation number:
- 0: User-input FC and FPOW even for single-phase flow
- 1: McAdam's Homogeneous
- 2: Lockhart-Martinelli
- 3: Baroczy
- 4: Friedel
- 5: Whalley's recommendations
- 6: Flow regime based two–phase friction

UPF ..... (if IPDC isn't 0) upstream fraction of properties to use in friction pressure drop calculations.
HC ..... initial head coefficient (impressed pressure gradient)
FC ..... initial irrecoverable loss coefficient
FPOW ..... flowrate exponent in frictional pressure drop term: $FC \cdot FR \cdot |FR|^{FPOW}$
AC ..... initial recoverable loss coefficient
FG ..... initial phase generation coefficient, for twinned tubes
FD ..... initial interface friction coefficient, for twinned tubes

defaults (if not previously defined in a PA DEF subblock):

AF ..... -1.0 (nonpositive value signals that AF should be calculated by assuming a circular cross section of diameter DH. Otherwise, DH and AF are independent parameters.)
AFTH ..... AF
FK ..... zero
WRF ..... zero
IPDC ..... 1 (McAdam's Homogeneous); 6 (flow regime mapping) if twinned
UPF ..... 0.5 (Average of up and downstream properties)
HC ..... zero

FC ..... zero (calculated automatically if IPDC isn't 0)

FPOW ..... 1.0 (calculated automatically if IPDC isn't 0)

AC ..... zero

FG ..... zero

FD ..... zero (calculated automatically if twin paths are input and IPDC=6)

Guidance: While the parameters FC, FPOW, AC, FG, and FD are available for direct input, they are normally calculated automatically by various options and their presence can be ignored by casual users.

Guidance: If a frictional pressure drop model other than those available is desired, or if FC is to be zero (no frictional losses) or otherwise provided by user logic, set IPDC=0. If IPDC=0, then the input values of TLEN, WRF, UPF, and DH are ignored although TLEN and DH might still be used by HTN and HTNC ties. If IPDC=0, the user assumed control of both FC and FPOW.

Guidance: IPDC=6 should be used with twinned paths, otherwise the interface friction term FD will remain at the unrealistic (but legal) limit of zero.

Guidance: If twinned, the initial flowrate is assumed to be the total flowrate. It will be apportioned to each twin on the basis of initial upstream quality.

Guidance: If twinned, all other initial values apply to both twins. If FC, FPOW, AC, FG, or FD are being specified in FLOW DATA for twinned STUBEs, the user should consider modifying these values for each twin in OPERATIONS DATA prior to a solution routine call since these parameters will rarely be equal for both twins. In fact, the magnitude of FG and FD should normally differ by the ratio of densities, and FG should be of opposite sign for each phase.

Restriction: Phase suction options (STAT) are ignored in conjunction with twinned STUBEs (see also GOHOMO and GOSLIP support subroutines in Section 6.)

Restriction: If IPDC=0, all of FK, FC, and AC cannot be zero.

Examples:

```
PA CONN,1,2,3, DEV=STUBE, FK=2.5, TLEN=14.0, WRF=0.0
     AF=1.0, DH=0.1
PA CONN,44,55,66, FR=2.0, STAT=DLS, DEV=STUBE
     IPDC=2, UPF=0.5
PA CONN,212,2,12, TWIN = 10212 $ GENERATES TWO STUBES
     IPDC = 6
```

CAPIL Format in CONN Subblocks:

```
... DEV=CAPIL, RC=R, CFC=R [,XVH=R][,XVL=R]
```

where:

RC ..... effective two-dimensional capillary radius
CFC ..... capillary flow conductance through connector
XVH ..... capillary priming dryness, upper quality limit
XVL ..... capillary priming dryness, lower quality limit

defaults:

XVH 0.99
XVL 0.95

Examples:

```
PA CONN,10,1,15, DEV=CAPIL, RC=10.0E-6, CFC=1.0E-11
PA CONN,1,5,10, FR=0.05, DEV=CAPIL
     RC  = 5.0E-6
     CFC = 1.7E-12
```

Guidance: Nonpositive values of RC will be interpreted as a perfect, infinite pressure difference capillary structure. This is useful in reducing the chance of deprime during steady-state runs.

Guidance: XVH and XVL represent the limits of a hysteresis cycle. An adjacent lump with a quality higher than the current value (XVH or XVL, depending on past history) is considered dry enough to permit the device to prime, or perhaps too dry to permit priming, depending on whether the lump has higher or lower pressure than the opposite lump.

Caution: A CAPIL connector does not by itself simulate dynamic wicking or capillary pumping; it can only prevent vapor flow when not deprimed. Refer to the CAPPMP macrocommand to satisfy capillary pumping simulation requirements.

Restrictions: STAT is overridden for this device, and so cannot be set independently.

LOSS Format In CONN Subblocks:

```
... DEV=LOSS, FK=R [,AF=R][,AFTH=R][,TPF=R]
```

where:

FK ..... K factor (head loss factor) through connector
AF ..... Effective flow area (basis of velocity)
AFTH ..... throat flow area for optional choking calculations
TPF ..... Two-phase factor (for nonhomogeneous losses)

defaults:

AF ..... AF from PA DEF subblock, or if no AF defined or if negative, calculated using DH from PA DEF subblock assuming circular cross section
AFTH ..... AF
TPF ..... 1.0 (homogeneous)

Restrictions: FK must be greater than zero.

Examples:

```
PA CONN,40,1,5, FR=22.0, DEV=LOSS, FK=2.5
PA CONN,1,1,2, DEV=LOSS, FK=1.0, AF=5.0E-4, TPF=1.5
```

LOSS2 Format in CONN Subblocks:

```
... DEV=LOSS2, FK=R [,FKB=R][,AF=R][,AFTH=R][,TPF=R]
```

where:

FK ..... K factor (head loss factor) through connector if flow is positive (forward)
FKB ..... K factor (head loss factor) through connector if flow is negative (backward)
AF ..... Effective flow area (basis of velocity)
AFTH ..... throat flow area for optional choking calculations
TPF ..... Two-phase factor (for nonhomogeneous losses)

defaults:

FKB ..... FK
AF ..... AF from PA DEF subblock, or if no AF defined or if negative, calculated using DH from PA DEF subblock assuming circular cross section
AFTH ..... AF
TPF ..... 1.0 (homogeneous)

Examples:

```
PA CONN,4,5,8, STAT=VS, DEV=LOSS2, FK=1.0, FKB=10.0
PA CONN,66,12,13, DEV=LOSS2, FK=12.0, AF=-1.0
     FKB=0.5, TPF=2.0
```

Restrictions: FK and FKB must be greater than zero.

CHKVLV Format:

```
... DEV=CHKVLV, FK=R [,AF=R][,AFTH=R][,TPF=R]
```

where:

FK ..... K factor (head loss factor) through open check valve (nonzero)
AF ..... Effective flow area (basis of velocity when open)
AFTH ..... throat flow area for optional choking calculations
TPF ..... Two-phase factor (for nonhomogeneous losses when open)

defaults:

AF ..... AF from PA DEF subblock, or if no AF defined or if negative, calculated using DH from PA DEF subblock assuming circular cross section
AFTH ..... AF
TPF ..... 1.0 (homogeneous)

Guidance: Check valve will close if the flow rate through the valve is negative. For positive flow, the device corresponds to a LOSS connector in every way.

Restriction: A valve cannot close when used in series with an MFRSET or VFRSET connector, and a junction cannot be surrounded only by closed valves and MFRSET/VFRSET connectors. FK must be greater than zero.

Examples:

```
PA CONN,1011,10,11, DEV=CHKVLV, FK=0.05, AF=5.0E-4
PA CONN,1213,12,13, FR=2.0,STAT=NORM, DEV=CHKVLV
     FK=1.0
```

CTLVLV Format:

```
... DEV=CTLVLV, FK=R [,AF=R][,AFTH=R][,TPF=R]
```

where:

FK ..... K factor (head loss factor) through open control valve (nonzero)
AF ..... Effective flow area (basis of velocity when open)
AFTH ..... throat flow area for optional choking calculations
TPF ..... Two-phase factor (for nonhomogeneous losses when open)

defaults:

AF ..... AF from PA DEF subblock, or if no AF defined or if negative, calculated using DH from PA DEF subblock assuming circular cross section
AFTH ..... AF
TPF ..... 1.0 (homogeneous)

Guidance: Control valve will close if the FK is negative. For positive FK, the device corresponds to a LOSS connector in every way. It is the user's responsibility to manipulate FK (and/or AF if desired) within user logic blocks according to the desired control algorithm. Note that the control algorithm should mimic reality; FK changes slowly and gradually.

Restriction: A valve cannot close when used in series with an MFRSET or VFRSET connector or an operating capillary device, and a junction cannot be surrounded only by closed valves and MFRSET/VFRSET connectors.

Examples:

```
PA CONN,12,13,14, DEV=CTLVLV, FK=-1.0
PA CONN,112,44,19, FR=0.01, DEV=CTLVLV, AF=-1.0
     FK=0.01, TPF=1.1
```

MFRSET Format in CONN subblocks:

```
... DEV=MFRSET [,SMFR=R]
```

where:

SMFR ..... Mass flow rate to maintain.

defaults:

SMFR ..... FR from PA DEF subblock, or if no default FR defined, zero. Note: to be consistent with other connectors, an FR value defined earlier in this subblock is taken to be an initial value unrelated to SMFR.

Restriction: *Do not use more than one in series.* In general, do not put this device in any situation where maintaining the given flow rate contradicts a condition such as a closed valve or a primed capillary device.

Examples:

```
PA CONN,6,5,4, DEV=MFRSET
PA CONN,12,1,118, FR=12.0, DEV=MFRSET
PA CONN,12,1,118, DEV=MFRSET, SMFR=12.0
```

VFRSET Format in CONN subblocks:

```
... DEV=VFRSET, SVFR=R
```

where:

SVFR ..... Volumetric flow rate to maintain

defaults:

SVFR ..... zero

Restriction: *Do not put more than one in series unless compressible (at least part vapor) tanks will* always *be present between them and FASTIC is not used.* In general, do not put this device in any situation where maintaining the given volumetric flow rate contradicts a condition such as a closed valve or a primed capillary device.

Examples:

```
PA CONN,118,119,120, DEV=VFRSET, SVFR=0.5
PA CONN,115,16,17, STAT=LS, DEV=VFRSET
     SVFR=12.0/62.4
```

PUMP Format in CONN subblocks:

```
... DEV=PUMP, HG=A [,DEGVF=A]
```

where:

HG ..... Doublet array containing the pump head curve: head (length) as a function of G (volumetric flow rate). *G must increase monotonically, and Head must decrease monotonically.* G1,H1,G2,H2 ...

DEGVF .... Doublet array containing the head degradation factor as a function of void fraction. $0.0 \preceq DEG \prec 1.0$; VF1,DEG1,VF2,DEG2 ...

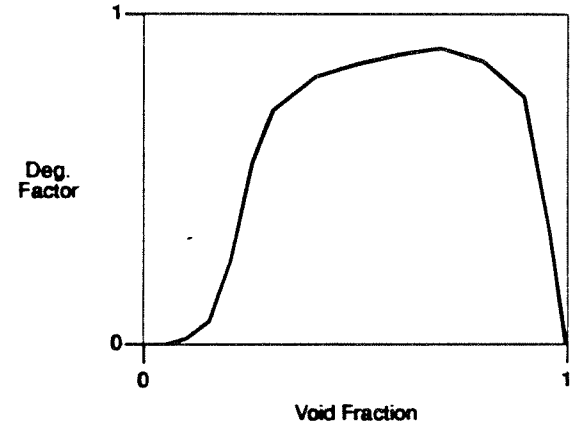
defaults:

DEGVF .... Uses a "standard" curve that is stored internally (Figure 3–9)

Guidance: The independent variable in each array should extend over the range of values anticipated in the analysis. Otherwise, performance will be extrapolated to meet the actual range requirements.

Examples:

```
PA CONN, 10,1,2, FR=10.0, DEV=PUMP, HG=1, DEGVF=2
PA CONN,1,111,112, STAT=DLS, DEV=PUMP, HG=114
```



Void Fraction	Deg. Factor
0.0	0.0
0.05	0.0
0.10	0.02
0.15	0.07
0.20	0.25
0.25	0.55
0.30	0.71
0.40	0.81
0.50	0.85
0.60	0.88
0.70	0.90
0.80	0.86
0.90	0.75
0.96	0.35
1.0	0.0

Figure 3–9 Default Two-Phase Pump Degradation Curve

VPUMP Format in CONN subblocks:

```
... DEV=VPUMP, HG=A, RSPD=R [,SPD=R][,DEGVF=A]
```

where:

HG ..... Doublet array containing the pump head curve: head (length) as a function of G (volumetric flow rate). *G must increase monotonically, and Head must decrease monotonically.* G1,H1,G2,H2 ...

RSPD ..... Reference pump speed in any user selected units

SPD ..... Initial pump speed in same units as RSPD

DEGVF .... Doublet array containing the head degradation factor as a function of void fraction. $0.0 \preceq DEG \prec 1.0$; VF1,DEG1,VF2,DEG2 ...

defaults:

SPD ..... RSPD

DEGVF .... Uses a "standard" curve that is stored internally (Figure 3–9)

Guidance: The independent variable in each array should extend over the range of values anticipated in the analysis. Otherwise, performance will be extrapolated to meet the actual range requirements. Note that SPD=0 is equivalent to a closed valve to be consistent with the pump similarity rules.

Examples:

```
PA CONN,1,2,3, FR=10.0, DEV=VPUMP, HG=1
     RSPD      = 4000.0
     SPD       = 8000.0
PA CONN,11,12,13, DEV=VPUMP, HG=114, RSPD=1.0
     DEGVF=2
```

3.11.2.5 Thermal Heat Transfer Tie (T Subblocks)

This section describes the formats used to input a *tie*, a special conductor-like element used to link fluid submodels with thermal submodels. Specifically, a tie always extends between a FLUINT lump and a SINDA node to permit energy interchange. In general, the lump represents the fluid within a container or pipe segment, and the node represents the inner wall of the container. With minor exceptions, any number of ties (up to the default size limit of 6000—see Section 1.5) may exist between any lump and any node. In fact, unlike any other SINDA/FLUINT element, ties can be dynamically moved to and from any other lump or node using the PUTTIE routine (see Section 6).

All ties have a translatable UA value analogous to the thermal conductor G value. This "conductance" is either calculated by the program or supplied by the user, depending on the option used. The tie conductance is *normally* defined by: $QTIE_{tie} = UA_{tie} \cdot (T_{node} - TL_{lump})$. The QTIE value is subtracted from the nodal source term for all ties on that node. (Actually, during steady-states the QTIE is summed but ignored by the node, which treats the adjacent lump as if it were another node.) Similarly, the QTIE value is summed into the lump source term QDOT for all ties on that lump, replacing any user inputs for QDOT.

Analogous to path duplication factors, tie duplication factors DUPL and DUPN are available. These values may be used to shut off ties (by setting them to zero), to avoid or to accommodate dual one-way SINDA conductors, etc. DUPN is the factor applied to the node end of the tie, and DUPL is the factor applied to the lump end. The node "sees" DUPN such ties, and the lump "sees" DUPL such ties. Thus, the value summed into the lump QDOT is actually DUPL•QTIE, while the value subtracted from the nodal Q is actually DUPN•QTIE.

There are six options available:

HTN ..... Convection heat transfer, involving one path or one pair of twinned paths
HTNC ..... Convection heat transfer, involving two paths or two pairs of twinned paths (centered lump)
HTNS ..... Segment–oriented convection heat transfer over one path
HTU ..... User-described heat transfer
HTUS ..... Segment–oriented user–described heat transfer over one path
HTM ..... Macro-specific heat transfer (CAPPMP and MGSEG)

Because the last option is only pertinent to macrocommands, it will be described later in that section.

The remainder of the options may be classified in two ways. With HTN, HTNC, and HTNS ties, the UA value is calculated automatically by the program according to convective heat transfer in a duct. With HTU and HTUS ties, the user provides the UA coefficient.

The second distinction is more subtle. HTN, HTNC, and HTU ties calculate heat transfer assuming that the lump represents the bulk fluid properties (according to the above equations), and that the node represents the average wall temperature. This implicitly assumes downstream–weighted heat transfer, since the lump state changes according to the heat transfer rate, and only that state is used in calculating heat transfer. Adjacent paths (tubes and STUBE connectors) are used only to provide flowrate and heat transfer area data; whether they are upstream or downstream of the lump is irrelevant.

While these lumped parameter methods are robust and are recommended for two–phase flows, they tend to underpredict heat transfer rates slightly for single phase flows, especially when the resolution is too coarse. Also, they tend to conflict with the way many engineers treat a single–phase heat transfer problem: as a constant wall temperature over a segment that has distinct inlet and outlet states.

The second type of tie, HTNS and HTUS, is called a *segment* tie. Segment ties treat the heat transfer problem differently than do traditional ties. Instead of being lump–oriented, they are path–oriented. The node is assumed to represent the average wall temperature over the length of the path, the lumps on either end of that path are assumed to represent the upstream and downstream states of that segment. Unless the user specifies otherwise, *these ties will automatically jump to the lump that is currently downstream of the specified path when flows reverse.*

For segment ties, $QTIE = (Q_{up} - Q_{down})/\ln(Q_{up}/Q_{down})$, where $Q_{up} = U_{up}*A*(T_{node} - T_{lump,up})$ and $Q_{down} = U_{down}*A*(T_{node} - T_{lump,down})$ by default, representing an logarithmic average of the inlet and outlet heat transfer rates. For single–phase flows, U is relatively constant and this solution is equivalent to a log–mean temperature difference (LMTD) solution. If either lump is two–phase, or if flows reverse and the "STAY" option (described later) has been used, the ties revert to downstream–weighted (HTU and HTN) behavior: $QTIE = Q_{down}$. For single–phase flows, if $|Q_{down}| > |Q_{up}|$ then an arithmetic average is used. In the special case of HTUS ties in single–phase flows, the QTIE value will usually exceed $QTIE_{tie} = UA_{tie}*(T_{node} - TL_{lump})$ because of the effects of the upstream lump.

Guidance: ***While HTNS ties are preferred in the limit of single–phase steady flows, HTNC ties are recommended for two–phase unsteady flows or where flow conditions are not known a priori.***

Guidance: If the thermal submodel containing the node is not currently active, the tie will be adiabatic (QTIE=0.0).

Restriction: The user forfeits his ability to specify QDOT for a lump if any ties are placed on that lump. If a tie is moved (via PUTTIE), the user regains the ability to specify that QDOT value, which would otherwise be reset to zero.

HTN Subblock Format:

```
T HTN,tid,lid,nid,pid[,f][,DUPN=R][,DUPL=R]
```

where:

`tid`	tie id
`lid`	lump id with which heat will be transferred
`nid`	node id with which heat will be transferred. This must be input in the format tsmn.nodnam, where tsmn is the thermal submodel name containing the node named nodnam
`pid`	path id of the path to use for flow rate and geometry (either tube or STUBE connector). Alternatively, the path id of the primary path of a pair of twins (the involvement of the secondary path is implied).
`f`	fraction of pid wetted area attributed to tie
`DUPN`	node duplication factor (number of this tie seen by nid)
`DUPL`	lump duplication factor (number of this tie seen by lid)

defaults:

`f`	1.0
`DUPN`	1.0
`DUPL`	1.0

Guidance: This option specifies the automatic calculation of convective heat exchange between a fluid lump and a thermal node that represents the container wall. A standard set of convection correlations are used, including correlations for boiling and condensing (see Appendix B). The path named pid will be used to provide geometry (diameter, perimeter, flow area, length, and heat transfer area) and flow rate information to the correlations. If a twinned pair of paths is used, only the ID of the primary path need be input—the involvement of the secondary path is implied. Heat transfer to a constant wall temperature may be simulated if nid is a boundary node. Alternatively, a fluid-dominated thermal boundary condition may be simulated if lid is a plenum. This option may be used to specify one heat exchanger segment where the mass and volume of the segment are contained in lid, and the length and cross-section of the same segment are described by pid. lid may be up or downstream of pid, depending on modeling preferences. The area fraction is applied to the final UA value, and hence may be used to model non-uniform temperatures, fouling, etc.

Restrictions: *lid and pid must be separately defined in **this** fluid submodel. nid must be defined in a thermal submodel.* The QDOT for lid will be calculated automatically during execution, and will be summed into the Q for nid. If f=0.0, the tie will be adiabatic.

Restrictions: *pid cannot reference a secondary twin, or a path other than a tube or STUBE.*

Examples:

```
T HTN,10,10,RIBS.2014,33,  DUPL=2.0, DUPN=2.0
T HTN,22,222,SLAW.2,2203,0.5
```

HTNC Subblock Format:

```
T HTNC,tid,lid,nid,pid,f1,p2id[,f2][,DUPN=R][,DUPL=R]
```

where:

`tid`	tie id
`lid`	lump id with which heat will be transferred
`nid`	node id with which heat will be transferred. This must be input in the format tsmn.nodnam, where tsmn is the thermal submodel name containing the node named nodnam
`pid`	path id of first path to use for flow rate and geometry (either tube or STUBE connector). Alternatively, the path id of the primary path of a first pair of twins (the involvement of the secondary path is implied).
`f1`	fraction of pid wetted area attributed to tie
`p2id`	path id of second path to use for flow rate and geometry (either tube or STUBE connector). Alternatively, the path id of the primary path of a second pair of twins (the involvement of the secondary path is implied).
`f2`	fraction of p2id wetted area attributed to tie
`DUPN`	node duplication factor (number of this tie seen by nid)
`DUPL`	lump duplication factor (number of this tie seen by lid)

defaults:

`f2`	0.5
`DUPN`	1.0
`DUPL`	1.0

Guidance: This option specifies the automatic calculation of heat exchange between a fluid lump and a thermal node that represents the container wall. A standard set of convection correlations are used, including correlations for boiling and condensing. The paths named p1id and p2id will be used to provide geometry (diameter, perimeter, flow area, length, and heat transfer area) and flow rate information to the correlations. If a twinned pair of paths is used, only the ID of the primary path need be input—the involvement of the secondary path is implied. Both, either, or neither of p1id and p2id may represent twins; any combination is allowed. Heat transfer to a constant wall temperature may be simulated if nid is a boundary node. Alternatively, a fluid-dominated thermal boundary condition may be simulated if lid is a plenum. This option may be used to specify one heat exchanger segment where the mass and volume of the segment are contained in lid. The segment is assumed to span f1 of p1id, and f2 of p2id. Thus, lid is assumed to be "centered" between p1id and p2id if f1=f2=0.5, and this option is the same as HTN if f1=1.0 and f2=0.0. Note that f1 and f2 need not sum to one. The area fractions are applied to the final UA value, and hence may be used to model nonuniform temperatures, fouling, etc.

Restrictions: *lid, p1id, and p2id must be separately defined in* ***this*** *fluid submodel. nid must be defined in a thermal submodel.* The QDOT for lid will be calculated automatically during execution, and will be summed into the Q for nid. If f1=f2=0.0, then the tie heat rate QTIE will be zero.

Restrictions: *p1id and p2id cannot reference a secondary twin, or a path other than a tube or STUBE.*

Examples:

```
T HTNC,10,10,RIBS.2014, 33,0.5,        34,1.0
T HTNC,22,222,SLAW.2
               2203,1.0
               2204,1.0
               DUPN    = 0.0   $ Node is unaware of tie
T HTNC,144,101,BEANS.24, 633,0.5, 366
```

HTU Subblock Format:

```
T HTU,tid,lid,nid,UA=R[,DUPN=R][,DUPL=R]
```

where:

tid	tie id
lid	lump id with which heat will be transferred
nid	node id with which heat will be transferred. This must be input in the format tsmn.nodnam, where tsmn is the thermal submodel name containing the node named nodnam
UA	conductance of the tie. The conductance must be positive or zero.
DUPN	node duplication factor (number of this tie seen by nid)
DUPL	lump duplication factor (number of this tie seen by lid)

defaults:

DUPN	1.0
DUPL	1.0

Guidance: This option allows the user to define the overall heat transfer between the lump and the node according for almost any phenomena. Note that no path is required. The UA value is held constant over each solution interval.

Restrictions: *lid must be separately defined in* ***this*** *fluid submodel. nid must be defined in a thermal submodel.* The QDOT for lid will be calculated automatically during execution, and will be summed into the Q for nid. If UA=0.0, then the tie heat rate QTIE will be zero.

Examples:

```
T HTU,10,10,RIBS.2014, UA=2.0
T HTU,22,222,SLAW.2,2203,
     UA = 0.0
```

HTNS Subblock Format:

```
T HTNS,tid,pid,nid,[,f][,DUPN=R][,DUPL=R][,STAY]
```

where:

`tid` ..... tie id

`pid` ..... path id of the path representing the segment, to be used for flow rate and geometry (either tube or STUBE connector). Alternatively, the path id of the primary path of a pair of twins (the involvement of the secondary path is implied). The tie will attach to the lump downstream of this path, moving as needed if flowrates reverse unless the STAY option is in effect (see below)

`nid` ..... node id with which heat will be transferred. This must be input in the format tsmn.nodnam, where tsmn is the thermal submodel name containing the node named nodnam

`f` ..... fraction of pid wetted area attributed to tie

`DUPN` ..... node duplication factor (number of this tie seen by nid)

`DUPL` ..... lump duplication factor (number of this tie seen by lid)

`STAY` ..... if present, indicates that the tie should remain on the defined downstream end of pid even if the flowrate reverses; otherwise the tie will automatically jump to the current downstream end

defaults:

`f` 1.0
`DUPN` 1.0
`DUPL` 1.0

Guidance: This option specifies the automatic calculation of convective heat exchange between a fluid heat exchanger segment and a thermal node that represents the average container wall temperature over the segment. The heat will be added or subtracted to the lump instantaneously downstream of pid, and this heat will move to the opposite lump if the flowrate in pid reverses and if 'STAY' has not been specified. A standard set of convection correlations are used, including correlations for boiling and condensing. The path named pid will be used to provide geometry (diameter, perimeter, flow area, length, and heat transfer area) and flow rate information to the correlations. If a twinned pair of paths is used, only the ID of the primary path need be input—the involvement of the secondary path is implied. The area fraction is applied to the final UA value, and hence may be used to model nonuniform temperatures, fouling, etc. A logarithmic average of the up– and downstream heat transfer rates is used unless either lump becomes two–phase or the flowrate has reversed and the 'STAY' command has been issued, in which case the heat rate is calculated as if this were an HTN tie.

Restrictions: *pid must be separately defined in **this** fluid submodel. nid must be defined in a thermal submodel.* The QDOT for the lump downstream of pid will be calculated automatically during execution, and will be summed into the Q for nid. If f=0.0, the tie will be adiabatic.

Restrictions: *pid cannot reference a secondary twin or a path type other than a tube or STUBE.*

Warning: If 'STAY' is not specified, then the tie can jump to either lump, and the DUPI and DUPJ options have the same magnifying effect on heat transfer area as with HTN and HTNC ties. An initial warning is produced if DUPI is not equal to DUPJ.

Examples:

```
T HTNS,10,33,RIBS.2014,  DUPL=2.0,STAY, DUPN=2.0
T HTNS,11,33,ROLLS.1133,0.75
T HTNS,22,222,SLAW.2, STAY
```

HTUS Subblock Format:

```
T HTUS,tid,pid,nid,UA=R[,DUPN=R][,DUPL=R][,STAY]
```

where:

tid	tie id
pid	path id of the path (any type) representing the segment. For twinned paths, pid should reference the primary path of the pair (the involvement of the secondary path is implied). The tie will attach to the lump downstream of this path, moving as needed if flowrates reverse unless the STAY option is in effect (see below).
nid	node id with which heat will be transferred. This must be input in the format tsmn.nodnam, where tsmn is the thermal submodel name containing the node named nodnam
UA	conductance of the tie. The conductance must be positive or zero.
DUPN	node duplication factor (number of this tie seen by nid)
DUPL	lump duplication factor (number of this tie seen by lid)
STAY	if present, indicates that the tie should remain on the defined downstream end of pid even if the flowrate reverses; otherwise the tie will automatically jump to the current downstream end.

defaults:

DUPN	1.0
DUPL	1.0

Guidance: This option allows the user to define the overall heat transfer between fluid heat exchanger segment and a thermal node that represents the average container wall temperature over the segment. The heat will be added or subtracted to the lump instantaneously downstream of pid, and this heat will move to the opposite lump if the flowrate in pid reverses and if 'STAY' has not been specified. The path pid can be of any type. The UA value is held constant over each solution interval. A logarithmic average of the up– and downstream heat transfer rates is used unless either lump becomes two–phase or the flowrate has reversed and the 'STAY' command has been issued, in which case the heat rate is calculated as if this were an HTU tie.

Note: The heat rate through the tie (QTIE) will normally exceed UA*ΔT for HTUS ties because of upstream gradients.

Restrictions: *pid must be separately defined in **this** fluid submodel, and cannot be a secondary twin. nid must be defined in a thermal submodel.* The QDOT for the lump downstream of pid will be calculated automatically during execution, and will be summed into the Q for nid. If UA=0.0, then the tie heat rate QTIE will be zero.

Examples:

```
T HTUS,10,10,RIBS.2014, STAY, UA=2.0
T HTUS,22,2203,SLAW.2, UA = 0.0
```

3.11.2.6 Element Generation Macros/Component Models (M Subblocks)

This section describes the formats used for macrocommands (macros). There are two basic types of such commands. The first type (element generation) is used to generate several related elements with a single command, and the number of such elements is variable. The second type (component model) is used to input a FLUINT model of a specific component that requires more than one element to model, but the number of such elements is fixed and/or they are of different types. In total, there are seven options available.

Element generation commands:

GENLU ..... To generate several lumps of the same type
GENPA..... To generate several paths of the same type, or several pairs of twinned paths of the same type
GENT ..... To generate several ties of the same type, or to add a row of ties to an HX or LINE duct model

Component models:

LINE..... To generate a string of lumps and paths representing an adiabatic duct
HX..... To generate a string of lumps, paths, and ties representing a diabatic duct (heat exchanger segment)
CAPPMP .... To input a model of a generic capillary pump

At first, the LINE and HX macros may seem more like element generation commands rather than component models. However, there are important differences between lumps and paths generated or input separately and those generated in order to represent a duct with internal variations in properties due to heat transfer or side flow passages. The difference is that FLUINT assigns no higher level meaning to a simple, perhaps chance catenation of lumps and paths, whereas a LINE or HX macro is known to represent a continuous flow passage. Additional terms (such as spatial accelerations due to velocity gradients) and methods are applied to LINE and HX macros that are not applied to elements that are input separately. Therefore, ***the analyst should use these duct macros whenever modeling an axially subdivided duct or other continuous flow passage.***

Guidance: No device is defined for connectors included in CAPPMP models. Although NULL will appear in output as the device model, the user should treat such connectors as if they were nonNULL connectors, and should not alter their GK or HK values.

Restriction: For all of the following element generation macros, *in order to use increments the starting value must be explicitly stated within the macro subblock* (i.e. starting values cannot be set in a DEF subblock).

Lump generator format:

```
M GENLU,mid,type,ilid[,N=I][,LUINC=I][,VOL=R]
      [,TL=R][,XL=R][,PL(!)=R][,VDOT=R][,QDOT=R]
      [,TLINC=R][,PLINC=R][,XLINC=R][,VOLINC=R]
      [,QDINC=R][,VDINC=R][,COMP=R][,CX=R][,CXINC=R]
      [,CY=R][,CYINC=R][,CZ=R][,CZINC=R]
```

where:

`mid` ..... macro identifier
`type` ..... TANK, JUNC, or PLEN
`ilid` ..... ID of first lump
`N` ..... number of lumps to generate
`LUINC` .... lump ID increment
`VOL` ..... initial volume of first tank (ignored if not tank)
`TL` ..... initial temperature
`XL` ..... initial quality
`PL` ..... initial pressure, PL! means PL has priority over TL
`VDOT` ..... initial time rate of change of tank volume (ignored if not tank)
`QDOT` ..... initial rate of heat input (ignored if plenum)
`TLINC` .... lump TL increment
`PLINC` .... lump PL increment
`XLINC` .... lump XL increment
`VOLINC` ... tank VOL increment (ignored if not tank)
`QDINC` .... lump QDOT increment (ignored if plenum)
`VDINC` .... tank VDOT increment (ignored if not tank)
`COMP` ..... tank compliance factor (ignored if not tank)
`CX` ..... location on the X axis
`CXINC` .... lump CX increment
`CY` ..... location on the Y axis
`CYINC` .... lump CY increment
`CZ` ..... location on the Z axis
`CZINC` .... lump CZ increment

defaults (if not previously defined in an LU DEF subblock):

```
N ......... 1
LUINC .... 1
XL ........ 0.0
VDOT ..... 0.0
QDOT ..... 0.0 (Note: QDOT will be calculated automatically if a this lump is tied)
TLINC .... 0.0
PLINC .... 0.0
XLINC .... 0.0
VOLINC ... 0.0
QDINC .... 0.0
VDINC .... 0.0
COMP ..... 0.0
CX ........ 0.0
CXINC .... 0.0
CY ........ 0.0
CYINC .... 0.0
CZ ........ 0.0
CZINC .... 0.0
```

Guidance: Values not applicable to the desired lump type are ignored (e.g., VDOT and VDINC for junctions). LU DEF subblocks can be used prior to a GENLU call if desired.

Example:

```
M GENLU,213,TANK,1,N=10, VOL=1.0, XL=0.0, TL=500.0,
     TLINC=10.0,PL=14.7, PLINC=0.1, COMP = 0.01/14.7
```

Path generator format:

```
M GENPA,mid,type,ipid,illid,il2id[,N=I][,PAINC=I]
      [,L1INC=I][,L2INC=I]
      [,FR=R][,FRINC=R][,STAT=C][,TWIN=I]
      [,DUPI=R][,DUPJ=R][,DUP=R]
      further path specifications ...
```

where:

`mid` ..... macro identifier
`type` ..... TUBE or CONN
`ipid` ..... ID of first path
`illid` .... ID of first upstream lump
`il2id` .... ID of first downstream lump
`N` ..... number of paths to create
`PAINC` .... path ID increment (also applies to tid if pairs of twins are generated)
`L1INC` .... upstream lump ID increment
`L2INC` .... downstream lump ID increment
`FR` ..... initial flow rate
`FRINC` .... increment of initial flow rate
`STAT` ..... phase suction status flag:
- NORM: normal (extracts both phases from upstream lump)
- LS: Liquid suction from defined upstream end only
- VS: Vapor suction from defined upstream end only
- DLS: Liquid suction from either end
- DVS: Vapor suction from either end
- RLS: Liquid suction from defined downstream end only
- RVS: Vapor suction from defined downstream end only

`TWIN` ..... ID of first secondary path, where ipid is this path's primary twin
`DUPI` ..... Upstream duplication factor for each path
`DUPJ` ..... Downstream duplication factor for each path
`DUP` ..... Duplication factor for each path (sets both DUPI and DUPJ)
`specs` .... remainder of TUBE or CONN specifications, as described in previous PA CONN and PA TUBE sections. For connectors, all device specific inputs must follow the DEV=C in the subblock.

defaults (if not previously defined in a PA DEF subblock):

```
N ......... 1
PAINC .... 1
L1INC .... 1
L2INC .... 1
FRINC .... 0.0
STAT ..... NORM
TWIN ..... no twins (single homogeneous paths generated)
DUPI ..... 1.0 (or value of DUP)
DUPJ ..... 1.0 (or value of DUP)
DUP ...... 1.0
specs .... see PA CONN and PA TUBE defaults
```

Guidance: Path specifications can vary greatly, especially for connector devices. The GENPA macro simply generates several of the same type path. As such, PA DEF subblocks can be used in conjunction with these macros as if the user had individually specified each path.

Guidance: The use of duplication factors with this macro is legal but should be used carefully. Duplication factors usually replace the need for this macro and vice versa.

Restriction: All device specific inputs for connectors must follow the DEV=C, as with individual connector subblocks, and these inputs must occur at the end of the subblock.

Example:

```
M GENPA,11,CONN,10,10,20, N=9, PAINC=1, L1INC=0,
     L2INC=0, FR=1.0, DEV=STUBE, DH=0.01,TLEN=2.0
     AF=1.0
```

Tie generator format:

The generic format for generating ties of all types is:

```
M GENT,mid,type,itid,ilpid,inid {,fs}{,ipid,f1,ip2id,f2}
     [,N=I][,LUINC=I][,PAINC=I][,PA2INC=I][,NINC=I]
     [,TIINC=I]{,UA=R}[,DUPN=R][,DUPL=R][,STAY]
```

where:

`mid`	macro identifier
`type`	HTN, HTNC, HTU, HTNS, HTUS, or HXC
`itid`	ID of first tie
`ilpid`	ID of first lump (HTN, HTNC, HTU, or HXC), or ID of the first segment path (HTNS, HTUS)
`inid`	ID of first node (smn.nid format)
`fs`	fraction of the segment paths' wetted area (HTNS only)
`ipid`	ID of first path (HTN, HTNC, and HXC only). Alternatively, the path id of the primary path of a first pair of twins (the involvement of the secondary path is implied).
`f1`	fraction of #1 paths' wetted area (HTN, HTNC, and HXC only). For the HXC option, this is the area fraction factor applied to *all* generated ties, similar to the AFRACT keyword in an HX macro.
`ip2id`	ID of first #2 path (HTNC only). Alternatively, the path id of the primary path of a second pair of twins (the involvement of the secondary path is implied).
`f2`	fraction of #2 paths' wetted area (HTNC only)
`N`	number of ties to generate
`LUINC`	lump ID increment (HTN, HTNC, HTU, and HXC only)
`PAINC`	path ID increment (HTN, HTNC, HTNS, and HXC only, #1 for HTNC)
`PA2INC` ...	#2 path ID increment (HTNC only)
`NINC`	Node ID increment (in same thermal submodel)
`TIINC`	tie ID increment
`UA`	initial tie conductance (HTU and HTUS only)
`DUPN`	node duplication factor (number of this tie seen by nid)
`DUPL`	lump duplication factor (number of this tie seen by lid)
`STAY`	if present, indicates that the tie should remain on the defined downstream end of pid even if the flowrate reverses; otherwise the tie will automatically jump to the current downstream end (HTNS and HTUS only)

defaults:

```
N ......... 1
LUINC .... 1
PAINC .... 1
PA2INC ... 1
NINC ..... 1
TIINC .... 1
fs ........ 1.0 for HTNS ties
f1 ........ 1.0 for HTN and HXC ties, required for HTNC ties
f2 ........ 0.5
DUPN ..... 1.0
DUPL ..... 1.0
```

Guidance: Inputs which may be required, depending on the type selected, appear between {} characters. Note that this macro generates ties between one fluid submodel and one thermal submodel only.

Guidance: The HTN, HTNC, HTNS, HTUS, and HTU options generate ties of those types. "HXC" does not correspond to a tie type. The GENT,HXC option generates an additional axial "row" of HTNC ties for a centered HX or LINE duct. (Up or downstream ducts can use the GENT,HTN option for additional axial rows.) This option differs from the GENT,HTNC option in that it automatically accounts for the area fraction distributions appropriate to centered duct models, as shown in the example below. Thus, even though the GENT,HXC option generates HTNC ties, its input requirements are more similar to the GENT,HTN option.

Restrictions: *Neither ilpid, ipid, ip2id, nor any increments can reference a secondary twin, or a path other than a tube or STUBE. (ilpid can reference any type of path for HTUS ties other than secondary twins.)*

Because most inputs vary by type of tie to be generated, the following type–dependent formats may be easier to use (where the use of ilpid has been replaced by ilid or ipid depending on whether the required input is a path or lump ID number):

```
M GENT,mid,HTU,itid,ilid,inid, UA = R
      [,N=I][,LUINC=I][,NINC=I]
      [,TIINC=I][,DUPN=R][,DUPL=R]


M GENT,mid,HTUS,itid,ipid,inid, UA = R
      [,N=I][,PAINC=I][,NINC=I]
      [,TIINC=I][,DUPN=R][,DUPL=R][,STAY]


M GENT,mid,HTN,itid,ilid,inid,ipid [,f1]
      [,N=I][,LUINC=I][,PAINC=I][,NINC=I]
      [,TIINC=I][,DUPN=R][,DUPL=R]
```

```
M GENT,mid,HTNS,itid,ipid,inid [,fs]
      [,N=I][,PAINC=I][,NINC=I]
      [,TIINC=I][,DUPN=R][,DUPL=R][,STAY]


M GENT,mid,HTNC,itid,ilid,inid,ipid,f1,ip2id [,f2]
      [,N=I][,LUINC=I][,PAINC=I][,PA2INC=I][,NINC=I]
      [,TIINC=I][,DUPN=R][,DUPL=R]


M GENT,mid,HXC,itid,ilid,inid,ipid [,f1]
      [,N=I][,LUINC=I][,PAINC=I][,NINC=I]
      [,TIINC=I][,DUPN=R][,DUPL=R]
```

Examples:

```
M GENT,100,HTN,10,10, BATT.10,        9,1.0
     N=10, TIINC=10, NINC=0
M GENT,120,HTNS,20,20, BATT.192, N=13, STAY
M GENT,101,HTNC,300,300, BATT.10,
                299,1.0,       301
M GENT,102,HXC,1,1,  WALL.1, 1, N=5
```

where the last example (HXC option) is equivalent to:

```
T HTNC, 1, 1, WALL.1,          1,1.0,  2,0.5
T HTNC, 2, 2, WALL.2,          2,0.5,  3,0.5
T HTNC, 3, 3, WALL.3,          3,0.5,  4,0.5
T HTNC, 4, 4, WALL.4,          4,0.5,  5,0.5
T HTNC, 5, 5, WALL.5,          5,0.5,  6,1.0
```

LINE (Untied Duct) Macro Format:

```
M LINE,mid,opt,ilid,ipid,llid{,l2id}[,NSEG=I]
      [,TLENT=R][,DHS=R][,AFS=R]
      [,AFTH=R][,FK=R][,FR=R]
      [,LU=C][,PA=C]
      [,LUINC=I][,TL=R][,PL(!)=R][,XL=R][,QDOT=R]
      [,TLINC=R][,PLINC=R][,XLINC=R][,QDINC=R]
      [,PAINC=I][,WRF=R][,HC=R][,FC=R][,FPOW=R][,AC=R]
      [,IPDC=I][,UPF=R][,COMP=R][,CX=R][,CXINC=R]
      [,CY=R][,CYINC=R][,CZ=R][,CZINC=R]
      [,TWIN=I][,AM=R][,FG=R][,FD=R]
      [,DUPI=R][,DUPJ=R][,DUP=R]
```

where:

`mid` ..... macro ID
`opt` ..... U, D, or C (Up, Down, or Center)
`ilid` ..... ID of first lump
`ipid` ..... ID of first path
`llid` ..... ID of lump to connect to (not generated,) at downstream end for opt=U, else at upstream end
`l2id` ..... ID of 2nd lump to connect to (not generated,) for opt=C only, at downstream end
`NSEG` ..... Number of segments (lumps) to generate (NSEG+1 paths will be generated if opt=C, twice that number if twinned)
`TLENT` .... total length of line to be generated
`DHS` ..... diameter of line
`AFS` ..... flow area of line
`AFTH` ..... throat flow area of each segment for optional choking calculations
`FK` ..... K–factor loss to apply to each segment
`FR` ..... flow rate through LINE
`LU` ..... lump type: TANK, JUNC, or PLEN
`PA` ..... path type: TUBE or STUBE (connector)
`LUINC` .... lump ID increment
`TL` ..... init. temp. of first lump
`PL` ..... init. pressure of first lump, PL! means PL has priority over TL
`XL` ..... init. quality of first lump
`QDOT` ..... init. heat rate of first lump
`TLINC` .... lump TL increment
`PLINC` .... lump PL increment
`XLINC` .... lump XL increment

QDINC .... lump QDOT increment
PAINC .... path ID increment (also applies to twin path itid)
WRF ..... wall roughness fraction
HC ..... init. head coef. for each path
FC ..... init. irrec. loss coef. for each path
FPOW ..... init. flowrate exponent in FC term for each path
AC ..... init. rec. loss coef. for each path
IPDC ..... press. drop correlation for paths
UPF ..... upwind fraction for each path
COMP ..... tank compliance factor (ignored if not tank)
CX ..... location on the X axis
CXINC .... lump CX increment
CY ..... location on the Y axis
CYINC .... lump CY increment
CZ ..... location on the Z axis
CZINC .... lump CZ increment
TWIN ..... unique identifier of initial secondary twin path to be generated (for slip flow modeling), where ipid is the corresponding primary twin
AM ..... initial added (virtual) mass coefficient, for twinned tubes
FG ..... initial phase generation coefficient, for twinned tubes
FD ..... initial interface friction coefficient, for twinned tubes
DUPI ..... total upstream duplication factor (applies only to upstream end of first path, ignored if opt=U)
DUPJ ..... total downstream duplication factor (applies only to downstream end of last path, ignored if opt=D)
DUP ..... Sets both DUPI and DUPJ (one of which will be ignored unless opt=C)

defaults (if not previously defined in a PA DEF or LU DEF subblock):

NSEG 1
AFS ..... negative (from DHS assuming circular x-section)
AFTH ..... AFS
FK ..... zero
LU ..... TANK
PA ..... TUBE
LUINC 1
XL 0.0
QDOT 0.0
TLINC 0.0
PLINC 0.0
XLINC 0.0
QDINC 0.0
PAINC 1
WRF 0.0
HC 0.0
FC ..... 0.0 (calculated automatically if IPDC isn't 0)
FPOW ..... 1.0 (calculated automatically if IPDC isn't 0)
AC ..... (calculated automatically)
IPDC ..... 1, or 6 if paths are twinned
UPF 0.5
COMP 0.0
CX 0.0
CXINC 0.0
CY 0.0
CYINC 0.0
CZ 0.0
CZINC 0.0
TWIN ..... no twins (single homogeneous paths created)
AM ..... zero (calculated automatically if twin tubes are input)
FG ..... zero (calculated automatically if twin paths are input)
FD ..... zero (calculated automatically if twin paths are input and IPDC=6)
DUPI ..... 1.0 (or DUP)
DUPJ ..... 1.0 (or DUP)
DUP 1.0

Guidance: A line is a series of paths and lumps that simulates a pipe, duct, or long passage. Lines can be composed of tanks, junctions, or plena, and tubes or STUBE connectors. If tanks are used, VDOT terms are assumed to be zero initially, and volumes are automatically calculated according to segment size and other options. If plena are used (perhaps for boundary conditions for a thermal model) QDOT and QDINC are ignored. PA DEF and LU DEF options apply to most inputs and should be used for clarity unless increment options are used.

Guidance: The configuration of each segment depends on the option used. If opt=U, NSEG segments of equal length are generated, with the lump upstream of the corresponding path. In this case, the user must name the lump at the downstream end of the line into which the last path will flow. If opt=D, NSEG segments of equal length will be generated, with the lump downstream of the corresponding path, and the user must name the lump at the upstream end of the line. If opt=C, NSEG lumps are generated, centered between NSEG+1 paths. In this case, all lumps are of equal size, but the first and last paths are one half the length of the remaining paths, and the user must name both the upstream and downstream lumps to which these half-sized paths will attach.

Guidance: The AC factor is calculated automatically for spatial accelerations within the duct due to heating or side flow. Note that rapid cooling or condensation or extreme suction can cause numerically destabilizing decelerations. Make sure all vapor-side losses (such as manifold entrances) are included as LOSS connectors or as FK factors applied at the entrance and exit. If twinned, the FG factor is also automatically calculated; applying nonzero FK disables slip flow options.

Caution: TLENT is the *total* length of the line to be generated, not the length of each segment. *Both TLENT and DHS are required inputs with no defaults.*

Restriction: Only tubes and STUBE connector devices may be used in LINE macro calls. Also, no heat transfer ties are generated, but these may be added via the GENT macro. The GENT,HTN is applicable if opt=U or D, else use GENT,HXC.

Example:

```
M LINE,1,U,2,20,1,NSEG=10, TLENT=10.0
     DHS=1.0/12.0,AFS=-1.0,FR=13.0
     LU=JUNC, PA=STUBE
     QDOT=0.0, QDINC=100.0
     TL=400, TLINC=10.0, UPF=1.0
```

HX (Tied Duct) Macro Format:

```
M HX,mid,opt,ilid,ipid,itid,inid,l1id{,l2id}[,NSEG=I]
      [,NINC=I][,TLENT=R][,DHS=R][,AFS=R]
      [,AFTH=R][,FK=R][,FR=R]
      [,LU=C][,PA=C]
      [,LUINC=I][,TL=R][,PL(!)=R][,XL=R]
      [,TLINC=R][,PLINC=R][,XLINC=R]
      [,PAINC=I][,WRF=R][,HC=R][,FC=R][,FPOW=R]
      [,AC=R][,IPDC=I][,UPF=R][,COMP=R][,CX=R]
      [,CXINC=R][,CY=R][,CYINC=R][,CZ=R][,CZINC=R]
      [,TWIN=I][,AM=R][,FG=R][,FD=R]
      [,DUPI=R][,DUPJ=R][,DUP=R][,DUPN=R][,DUPL=R]
      [,AFRACT=R][,TIINC=I][,STAY]
```

where:

- `mid` ..... macro ID
- `opt` ..... U, D, DS, or C (Up, Down, Down Segmented, or Center)
- `ilid` ..... ID of first lump
- `ipid` ..... ID of first path
- `itid` ..... ID of first tie
- `inid` ..... ID of first node (smn.id) (tied but not generated)
- `l1id` ..... ID of lump to connect to (not generated,) at downstream end for opt=U, else at upstream end
- `l2id` ..... ID of 2nd lump to connect to (not generated,) for opt=C only, at downstream end
- `NSEG` ..... Number of segments (lumps) to generate (NSEG+1 paths will be generated if opt=C, twice that number if twinned)
- `NINC` ..... node ID increment (in same thermal submodel as first node)
- `TLENT` .... total length of line to be generated
- `DHS` ..... diameter of line
- `AFS` ..... flow area of line
- `AFTH` ..... throat flow area of each segment for optional choking calculations
- `FK` ..... K-factor loss to apply to each segment
- `FR` ..... flow rate through LINE
- `LU` ..... lump type: TANK, JUNC, or PLEN
- `PA` ..... path type: TUBE or STUBE (connector)
- `LUINC` .... lump ID increment
- `TL` ..... init. temp. of first lump
- `PL` ..... init. pressure of first lump, PL! means pressure has priority over TL
- `XL` ..... init. quality of first lump

TLINC .... lump TL increment
PLINC .... lump PL increment
XLINC .... lump XL increment
PAINC .... path ID increment
WRF ..... wall roughness fraction
HC ..... init. head coef. for each path
FC ..... init. irrec. loss coef. for each path
FPOW ..... init. flowrate exponent in FC term for each path
AC ..... init. rec. loss coef. for each path
IPDC ..... press. drop correlation for paths
UPF ..... upwind fraction for each path
COMP ..... tank compliance factor (ignored if not tank)
CX ..... location on the X axis
CXINC .... lump CX increment
CY ..... location on the Y axis
CYINC .... lump CY increment
CZ ..... location on the Z axis
CZINC .... lump CZ increment
TWIN ..... unique identifier of initial secondary twin path to be generated (for slip flow modeling), where ipid is the corresponding primary twin
AM ..... initial added (virtual) mass coefficient, for twinned tubes
FG ..... initial phase generation coefficient, for twinned tubes
FD ..... initial interface friction coefficient, for twinned tubes
DUPI ..... total upstream duplication factor (applies only to upstream end of first path, ignored if opt=U)
DUPJ ..... total downstream duplication factor (applies only to downstream end of last path, ignored if opt=D or opt=DS)
DUP ..... Sets both DUPI and DUPJ (one of which will be ignored unless opt=C)
DUPN ..... node side tie duplication factor (applies only all ties)
DUPL ..... lump side tie duplication factor (applies only all ties)
AFRACT ... heat transfer area multiplier (ratio of heat transfer area to total wetted area)
TIINC .... tie ID increment
STAY ..... valid only if opt=DS. If present, indicates that the HTNS ties should remain on the defined downstream end of each segment even if the flow-rate reverses; otherwise the ties will automatically jump to the current downstream end

defaults (if not previously defined in a PA DEF or LU DEF subblock):

NSEG 1
NINC 1
AFS ..... negative (from DHS assuming circular x-section)
AFTH ..... AFS
FK ..... zero
LU ..... TANK
PA ..... TUBE
LUINC 1
XL 0.0
TLINC 0.0
PLINC 0.0
XLINC 0.0
PAINC 1
WRF 0.0
HC 0.0
FC ..... 0.0 (calculated automatically if IPDC isn't 0.0)
FPOW ..... 1.0 (calculated automatically if IPDC isn't 0.0)
AC ..... (calculated automatically)
IPDC ..... 1, or 6 if paths are twinned
UPF 0.5
COMP 0.0
CX 0.0
CXINC 0.0
CY 0.0
CYINC 0.0
CZ 0.0
CZINC 0.0
TWIN ..... no twins (single homogeneous paths created)
AM ..... zero (calculated automatically if twin tubes are input)
FG ..... zero (calculated automatically if twin paths are input)
FD ..... zero (calculated automatically if twin paths are input and IPDC=6)
DUPI ..... 1.0 (or DUP)
DUPJ ..... 1.0 (or DUP)
DUP 1.0
DUPN 1.0
DUPL 1.0
TIINC 1
AFRACT ... 1.0

Guidance: An HX command performs the same functions as a LINE command, except that ties are generated between each lump and a thermal node. HTN ties are generated for options U and D, HTNS ties are generated for option DS (which is otherwise identical to option D), and HTNC ties are generated for option C. An HX model is a series of paths and lumps that simulates a pipe or long passage with radial heat addition or rejection. HX segments can be composed of tanks, junctions, or plena, and tubes or STUBE connectors. If tanks are used, VDOT terms are assumed to be zero initially, and volumes are automatically calculated according to segment size and other options. PA DEF and LU DEF options apply to many inputs, and should be used when possible for clarity except when increment options are desired.

Guidance: The configuration of each segment depends on the option used. If opt=U, NSEG segments of equal length are generated, with the lump upstream of the corresponding path. In this case, the user must name the lump at the downstream end of the line into which the last path will flow. If opt=D or opt=DS, NSEG segments of equal length will be generated, with the lump downstream of the corresponding path, and the user must name the lump at the upstream end of the line. If opt=C, NSEG lumps are generated, centered between NSEG+1 paths. In this case, all lumps are of equal size, but the first and last paths are one half the length of the remaining paths, and the user must name both the upstream and downstream lumps to which these half-sized paths will attach. AFRACT is a multiplier that may be used to alter the area fractions of all ties. For circumferential wall gradients, additional axial "rows" of ties may be added to the duct via the GENT macro. Use GENT,HTN for opt=U or D, the GENT,HTNS for opt=DS, use GENT,HXC for opt=C.

Guidance: The AC factor is calculated automatically for spatial accelerations within the duct due to heating or side flow. Note that rapid cooling or condensation or extreme suction can cause numerically destabilizing decelerations. Make sure all vapor-side losses (such as manifold entrances) are included as LOSS connectors or as FK factors applied at the entrance and exit. If twinned, the FG factor is also automatically calculated; applying nonzero FK disables slip flow options.

Caution: TLENT is the *total* length of the line to be generated, not the length of each segment. *Both TLENT and DHS are required inputs.*

Caution: Use of opt=DS without the STAY command will cause the first tie to jump off the macro in the event that the flowrate in the first segment reverses.

Restriction: Only tubes and STUBE connector devices may be used in HX macro calls. Lump QDOTs are calculated automatically since heat transfer ties are generated for each lump.

Example:

```
M HX,1,C,2,20,2,BATT.3,1,101, NSEG=5, NINC=10
     TIINC=10, AFRACT=0.75
     TLENT=50.0, DHS=0.01, PA=TUBE, LU=TANK
     LUINC=1, PAINC=20, PL!=14.7, PLINC=-0.01
```

CAPPMP (Capillary Pump) Format:

```
M CAPPMP,mid,opt,ucid,jid,dcid,ulid,dlid,
      {,tid,tsmn.nid},RC=R,CFC=R[,XVH=R][,XVL=R]
      [,TL=R][,PL=R][,XL=R][,QDOT=R][,FR=R]
      [,DUPI=R][,DUPJ=R][,DUP=R][,DUPN=R][,DUPL=R]
      [,VAPOR=I]{,UA=R}
```

where:

mid	macro ID
opt	TIE ..... Generate a tie and provide a UA coefficient. NOTIE... Don't generate tie, provide QDOT instead.
ucid	ID of generated upstream connector
jid	ID of generated junction
dcid	ID of generated downstream connector
ulid	ID of nongenerated lump upstream of ucid
dlid	ID of nongenerated lump downstream of dcid
tid	ID of generated HTM tie
tsmn.nid	thermal node representing structure
RC	effective two-dimensional capillary radius
CFC	capillary flow conductance through entire passage
XVH	capillary priming dryness, upper quality limit
XVL	capillary priming dryness, lower quality limit
TL	init. temp. of junction jid
PL	init. pressure of junction jid
XL	init. quality of junction jid
QDOT	init. heat rate of junction jid (for NOTIE option)
FR	init. flow rate through CAPPMP.
DUPI	total upstream duplication factor (applies only to upstream end of ucid)
DUPJ	total downstream duplication factor (applies only to downstream end of dcid)
DUP	sets both DUPI and DUPJ
DUPN	node side tie duplication factor (for TIE option)
DUPL	lump side tie duplication factor (for TIE option)
VAPOR	"vapor" side: CAPPMP endpoint lump (ulid or dlid) at which heat transfer occurs (location of deprimed tie for TIE option)
UA	initial heat transfer conductance for the tie (required if opt=TIE)

defaults (if not previously set in a PA DEF or LU DEF subblock):

XVH 0.99
XVL 0.95
XL 0.0
QDOT 0.0
FR 0.0
DUPI ..... 1.0 (or DUP)
DUPJ ..... 1.0 (or DUP)
DUP 1.0
DUPN 1.0
DUPL 1.0
VAPOR .... dlid (default is valid positive flow for evaporation)

Guidance: Similar to a CAPIL connector, but generates a junction between two connectors. Pumping may occur according to the heat source/sink term on the junction: QDOT(jid). This source may either be input or calculated with an HTM tie given user-input UA, which operates similar to an HTU tie. Note that RC, CFC, XVH, and XVL are available for manipulation within user logic blocks as RC(ucid), CFC(ucid), XVH(ucid), and XVL(ucid). Nonpositive RC will be interpreted as a perfect, infinite pressure difference capillary structure. See also the SPRIME routine in Section 6.

Guidance: XVH and XVL represent the limits of a hysteresis cycle. An adjacent lump with a quality higher than the current value (XVH or XVL, depending on past history) is considered dry enough to permit the device to prime, or perhaps too dry to permit priming, depending on whether the lump has higher or lower pressure than the opposite lump.

Guidance: When a TIE option CAPPMP deprimes, the tie jumps to the designated vapor endpoint lump and becomes an HTU tie. It returns to the central junction and becomes an HTM tie if it reprimes. No analogous movements of the QDOT are possible for the NOTIE option. Refer to Section 4.7 for prime/deprime logic.

Caution: Changes to QDOT(jid) or UA during execution should be gradual except when changing to or from zero.

Restriction: No other ties may be placed on the central junction.

Examples:

```
M CAPPMP,10,NOTIE,12,2,23,1,3,XL=1.0
      CFC=1.0E-10, RC=1.0E-4
     QDOT=1000.0, FR=-100.0
M CAPPMP,11,TIE,22,223,23,1,3,223,WALL.220
     CFC=1.0E-10, RC=1.0E-4
     UA=10.0,  XVH = 0.85, XVL = 0.8
```

3.11.3 Sample Fluid Submodel Problem

In order to demonstrate input procedures and perhaps input style, a sample fluid submodel is given. A model description is given in Table 3–10, with the input file in Table 3–11. This sample corresponds to Sample Problem D in Appendix F. Refer to that volume for further descriptions.

The sample fluid submodel consists of the simple fluid loop pictured in Figure 3–10. The system consists of a pump circulating ammonia between a heat source and a heat sink. The system will be initialized as a two-phase transport loop, but FLUINT will continue to operate if the system becomes hard-filled or all vapor. A wicked bypass is being investigated to allow excess liquid to bypass the condenser. The element breakdown is described in Table 3–10. Note that the heat sink is a radiator modeled by a thermal network called RAD.

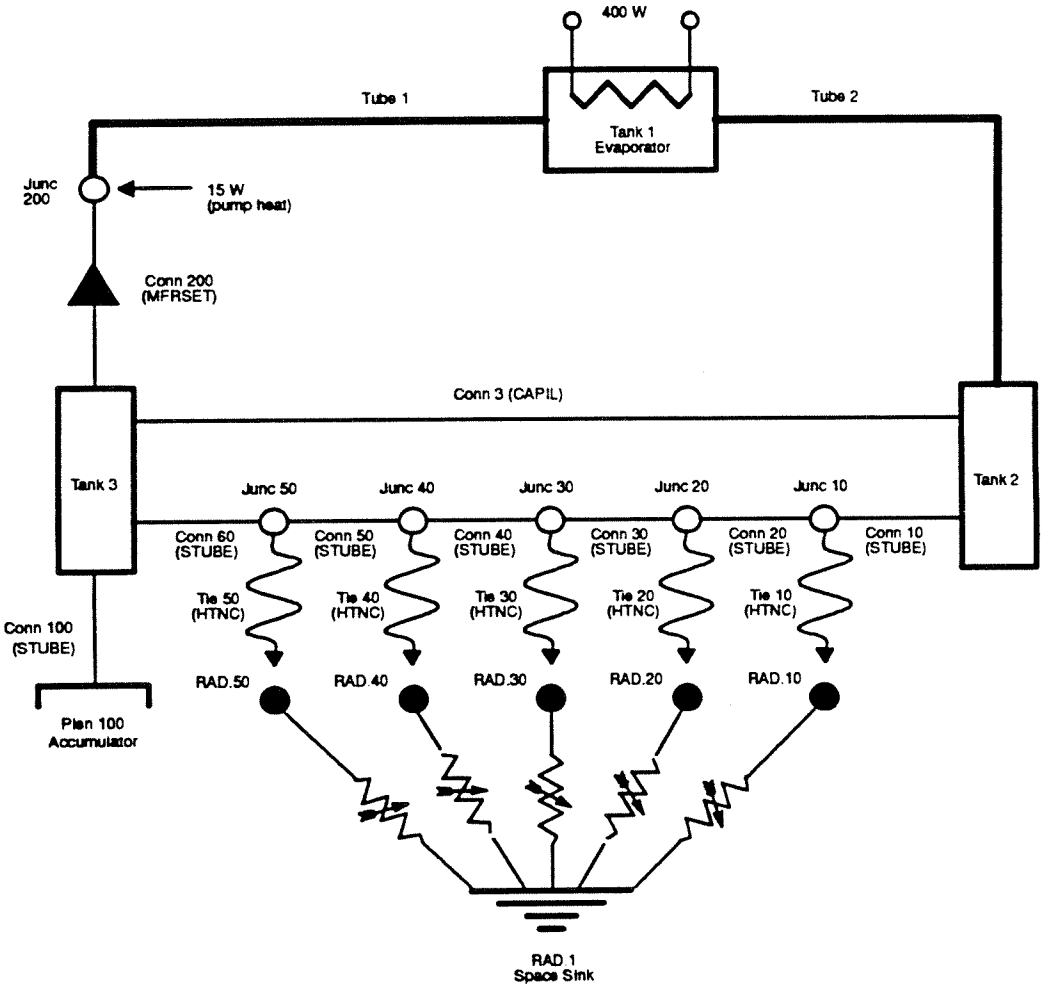


Figure 3–10 Schematic of FLUINT Sample Model

Table 3-10 Sample Model Description

ELEMENT	DESCRIPTION
TUBE 1	Liquid feed line from pump to evaporator.
TANK 1	Evaporator, containing all of the volume from the pump outlet to the condenser inlet.
TUBE 2	Vapor or two-phase line from evaporator to condenser.
TANK 2	Condenser inlet, containing half of the condenser volume.
JUNCTIONS 10-50	Heat exchanger segments that divide up the condenser into five parts.
STUBEs 10-60	Line segments in the condenser, corresponding to the above. #10 and #60 are half-length.
HTNC TIES 10-50	Tie LOOP.N invokes automatic convection heat transfer between junction LOOP.N and node RAD.N. Note centered differencing and area fractions mean #10 includes all of STUBE #10 and 1/2 of STUBE #20, etc.
TANK 3	Tee at the outlet of the condenser that connects the condenser, the bypass, the line to the accumulator, and the pump. Contains the second ond half of the condenser volume.
CAPIL 3	Wicked bypass connecting condenser inlet and outlet
STUBE 100	Small line to link the accumulator into the system.
PLENUM 100	Accumulator.
MFRSET 200	Idealized pump (constant mass flowrate).
JUNCTION 200	Between pump and delivery line (TUBE 1).
NODES 10-50	Diffusion nodes, each modeling 1/5 of the radiator.
NODE 1	Boundary node, representing effective radiation sink.

Table 3-11 FLOW DATA Block for Sample System

```
HEADER FLOW DATA,LOOP,FID=7717                  $ AMMONIA FILLED "LOOP"
C
C DEFINE MODEL DEFAULTS
C
LU DEF, TL = 300.0                              $ DEFAULT TEMPERATURE IS 300 K
           PLFACT     = 101325.0                $ PL MULTIPLIER (INPUT IN ATM)
           PL         = 6.0                     $ 6 ATM IS DEFAULT PRESSURE
           XL         = 0.0                     $ DEFAULT STATE IS LIQUID
PA DEF, FR = 5.0E-4                             $ FLOWRATE IS 0.5 GRAMS/SEC
           DH         = 0.008                   $ DEFAULT DIAMETER IS 8 mm
           TLEN       = 2.0                     $ DEFAULT DUCT LENGTH IS 2 METERS
           WRF        = 1.0E-6                  $ ASSUME FAIRLY SMOOTH PIPE
           UPF        = 0.5                     $ USE AVG. OF UP/DOWNSTREAM
           IPDC       = 4                       $ USE FRIEDEL'S CORRELATION
C
C START AT PUMP OUTLET, GO FLOWWISE AROUND LOOP
C
PA TUBE,1,200,1                                 $ TUBE FROM PUMP TO EVAPORATOR
C
LU TANK,1                                       $ EVAPORATOR CONTROL VOLUME
           VOL        = 4.0*0.008*0.008*0.25*3.1416
           QDOT       = 400.0
           XL         = 0.1                     $ NOTE PL WILL BE P AT SATURATION
C
PA TUBE,2,1,2                                   $ TUBE FROM EVAP TO SEPARATOR
C
LU TANK,2                                       $ SEPARATOR AND UPPER 1/2 OF COND
           VOL        = 2.5*0.008*0.008*0.25*3.1416
           XL         = 0.1                     $ NOTE PL WILL BE P AT SATURATION
C
C WICK A = 0.02, T = 1 cm.
```

Table 3-11 FLOW DATA Block for Sample System (concl)

```
C
PA CONN,3,2,3,          FR = 1.0E-4,          DEV = CAPIL
                        RC = 16.5E-6,         CFC = 2.6E-13*0.02/0.01
C
C START CONDENSER LINE (THE FOLLOWING 20 LINES WERE REPLACED BY
C AN M HX,C MACRO SUBBLOCK.  THIS INCLUDES DECELERATION OF VAPOR)
C USE CENTERED DIFFERENCING
C
CLU JUNC,10, XL         = 0.5                 $
CLU JUNC,20, XL         = 0.3                 $ NOTE QUALITY "GRADIENT"
CLU JUNC,30, XL         = 0.1                 $
CLU JUNC,40, TL         = 290.0               $ DEFAULT XL=0.0
CLU JUNC,50, TL         = 270.0               $ SUBCOOLED SECTION
C NOW DESCRIBE STUBE CONNECTORS THAT CORRESPOND TO ABOVE JUNCS
C
CPA DEF, TLEN = 1.0
CPA CONN,10,3,10,       DEV = STUBE,          TLEN = 0.50
CPA CONN,20,10,20,      DEV = STUBE
CPA CONN,30,20,30,      DEV = STUBE
CPA CONN,40,30,40,      DEV = STUBE
CPA CONN,50,40,50,      DEV = STUBE
CPA CONN,60,50,3,       DEV = STUBE,          TLEN = 0.50
C
C SET HEAT TRANSFER TIES IN CONDENSER
C NOTE CENTER PATHS ARE SHARED AMONG TIES
C
CT HTNC,10,10,RAD.10,  10,1.0,     20,0.5
CT HTNC,20,20,RAD.20,  20,0.5,     30,0.5
CT HTNC,30,30,RAD.30,  30,0.5,     40,0.5
CT HTNC,40,40,RAD.40,  40,0.5,     50,0.5
CT HTNC,50,50,RAD.50,  50,0.5,     60,1.0
C
C THE PREVIOUS 20 OR SO LINES WERE REPLACED WITH:
C
M HX,1,C,10,10,10,RAD.10,2,3
           NSEG = 5,            PA = STUBE,          LU = JUNC
           NINC = 10,           LUINC = 10,          PAINC = 10,TIINC = 10
           TLENT = 5.0,         DHS = 0.008,         FR = 4.0E-4
           XL = 0.4,            XLINC = -0.1,        UPF = 0.5
C
C NOTE THAT INITIAL CONDITIONS ARE DIFFERENT
C AND THAT AN HX MACRO ADDS SPATIAL ACCELERATION FACTORS AUTOMATICALLY
C
C FINISH LOOP TO PUMP, THEN ADD ACCUMULATOR
C
LU DEF, TL = 270.                            $ RESET DEFAULT TEMP. TO 270 K
C
C THE FOLLOWING TANK CONTAINS THE SECOND HALF OF THE
C VOLUME OF THE CONDENSER.
C
LU TANK,3, VOL = 2.5*0.008*0.008*0.25*3.1416
PA CONN,200,3,200, DEV = MFRSET              $ SIMPLEST PUMP MODEL
C                                            NOTE THE SMFR EQUALS THE DEF. FR.
C
LU JUNC,200                                  $ JUNC TO MATE PUMP TO EVAP. TUBE
           QDOT         = 15.0               $ 15 WATTS PUMP HEATING
PA CONN,100,3,100                            $ FLOW IS POSITIVE INTO ACCUM
           FR           = 0.0,      DEV      = STUBE
           TLEN         = 0.1,      DH       = 0.003
LU PLEN,100                                  $ ACCUMULATOR
```

3.11.4 Fluid Model Size Limits

The limits of FLUINT models are:

Maximum number of fluid submodels:	25
Maximum number of macros:	800
Maximum number of lumps:	4000
Maximum number of ties:	6000
Maximum number of paths:	4000
Max. no. of lumps per submodel:	1000
Max. no. of nonduplicated paths per lump:	25

The reader should note that the limit on lumps per submodel is only a suggested limit. However, the user would probably find it prohibitively expensive to have more than about 1000 tanks and junctions in any one submodel.* These "limits" are not immutable: they can be altered by resetting the appropriate value in the file PARAMETER.INC and recompiling the code. (See also Section 1.5.)

* The reason is that this number is one half of the order of the FLUINT solution matrix. Unlike iteratively solved thermal models (which take somewhat more than twice the computation time when the problem size is doubled, simultaneously solved FLUINT models take almost four times the computation time when the number of nonplenum lumps is doubled.

3.12 LOGIC BLOCKS

A logic block is a group of input records (delineated by a header record) that specifies a sequence of operations to be performed on the data input in the data blocks described in Sections 3.2 through 3.11 above. The general form of a logic block is as follows:

```
HEADER name-of-block,smn
          Statement-1
          Statement-2
                .
                .
                .
          Statement-n
```

where `name-of-block` can be any of the following:

```
OPERATIONS DATA*
VARIABLES 0
VARIABLES 1
VARIABLES 2
FLOGIC 0
FLOGIC 1
FLOGIC 2
OUTPUT CALLS
SUBROUTINE DATA*
```

Unlike data blocks, whose input formats are dictated by the type of block, all logic blocks are written in a Fortran-like language that is converted (or *translated*) into real Fortran. Almost all aforementioned element parameters (T, G, TL, FR, UA, etc.) can be translated, accessed, and perhaps modified within logic blocks. (See Section 3.12.2, Program Common, for a complete list.) Each logic block becomes a separate Fortran subroutine, with the exception of SUBROUTINE DATA, which is intended to contain several additional routines.

The subroutines resulting from the logic blocks fall into separate categories. The OPERATIONS DATA block is translated into subroutine OPER which is the user's main or driver routine. All of the calls to the SINDA solution routines (e.g., FWDBCK) should be made from this routine. It is also the appropriate place to initialize the values of any program variables that were not assigned values within the data blocks (e.g., NODE DATA).

The VARIABLES 0 blocks (one per thermal submodel) translate into a set subroutines called from subroutine VARBL0. The VARIABLES 1 blocks translate into a set of subroutines called from subroutine VARBL1, and the same is true for the VARIABLES 2 blocks in subroutine VARBL2. FLOGIC n blocks translate into routines called from FLOGIn. The set of OUTPUT CALLS blocks translate into subroutines called by subroutine OUTCAL. The VARBLn, FLOGIn, and OUTCAL subroutines are called from the solution routines at predefined points to update time-dependent thermal variables (VARBLO), temperature-dependent thermal variables (VARBL1), fluid variables (FLOGIn, as described below), and to perform program output operations (OUTCAL).

---

* Note smn is not allowed with these header names.

The SUBROUTINE DATA block is provided so that the user can add any number of arbitrarily named subroutines to the program that will be built from the logic blocks and the portions of the SINDA library that they invoke. These subroutines are useful where the same series of operations are required at a number of points in the other logic blocks. For example, the same specialized printout operations may be required in all or part of the individual OUTPUT CALLS blocks. SUBROUTINE DATA can also be used to replace standard or default subroutines with equivalent arguments but altered logical operations.

The logic blocks contain two types of statements: F-type Fortran statements and M-type Fortran statements*. An F-type statement is any valid Fortran statement *that requires no translation operations*. An M-type statement is similar to an F-type statement except that they may contain variables that require translation into real Fortran. These variables can include submodel name prefixes and actual element number suffixes such as: FRED.T10, meaning the temperature of node 10 in thermal submodel FRED. Such SINDA-specific subscripts are converted to the appropriate relative numbers that are used to find the variables in the program data storage. The two types of statements are illustrated below:

M - type statement:

```
CALL D11MDA(TIMEM,A200,A217,Q200,POD2.XK20)
```

F - type statement:

```
CALL D11MDA(TIMEM,A(21),A(38),Q(108),XK(5))
```

The distinction is *not* the presence or absence of parentheses. While F-type statements must use such parentheses while referring to an array, in M-type statements they are optional: A(200) is equivalent to A200. The real difference is that "A200" is a reference to a user array number 200, which is then translated into a cell in a single Fortran array (perhaps "A(21)") that contains all arrays input in all ARRAY DATA blocks. Normally, the user need not know into which array or cell location a particular variable is translated.

The following discussion of the SINDA logic blocks requires the introduction of several concepts if the prospective user is not familiar with Fortran programming techniques. This is true because the logic blocks are user's means of specifying the sequence of operations (mathematical and otherwise) to be performed by the computer in order to solve the user's problem, and, as such, they constitute a "computer program." Just as the network data blocks are used to tell the plug-and-grind computer *what* to plug, (the *data*), the operations blocks are used to tell it *how* to grind (the *logic*).

A general understanding of three key concepts is necessary in order to take full advantage of the versatility of the SINDA/FLUINT system. These concepts are (1) flow charts, (2) subroutine usage, and (3) program flow control. These concepts will not be discussed explicitly but they will serve to unify the discussion in the following sections. The discussion will show, by examples, how these concepts relate to the functional and operational aspects of the logic blocks.

A functional description of the logic blocks will be given in Section 3.12.1, and application guidelines will be presented in Section 3.12.2. The operational details of actually preparing SINDA/FLUINT logic records will be deferred until Section 3.12.3.

---

* Beginning with SINDA '85, SINDA-type statements have been replaced with M-type statements to improve preprocessor efficiency and to eliminate conversion errors.

3.12.1 Functional Description

After the preprocessor has processed the network data blocks and translated the logic blocks, the data is placed in the minimum required disk storage (thus releasing core memory for other uses) and the resulting subroutines are passed to the system FORTRAN compiler. Following compilation, the resulting program is loaded into core and executed as shown in the flow chart in Figure 3–11.

This simple flow chart reveals and implies several things:

1) The actions depicted in the flow chart take the place within the framework of a "main program" (the processor). Since this "main program" is fabricated entirely by the preprocessor, the actions therein occur *automatically* from the standpoint of the user (i.e., the user has no control over these actions).

2) The first action is the reading of the preprocessed data into core memory.

3) The final action is a transfer of control to subroutine OPER, which was formed from the user's OPERATIONS DATA block.

4) Clearly, the first opportunity for the user to specify operations which will lead to the solution of his problem occurs in the OPERATIONS DATA block. That is, if the user includes no operations in his block, then subroutine OPER will be empty (i.e., it will indicate that nothing is to be performed) and, for all intents and purposes, his program will do nothing. Conversely, exactly and only those operations included by the user in his OPERATIONS DATA block will appear and will be performed in subroutine OPER when it is called.

5) The basic flow chart includes no explicit reference to subroutines VARBLO, VARBL1, VARBL2, FLOGI0, FLOGI1, FLOGI2, or OUTCAL. These are called from solution routines, which are in turn normally called from OPER.

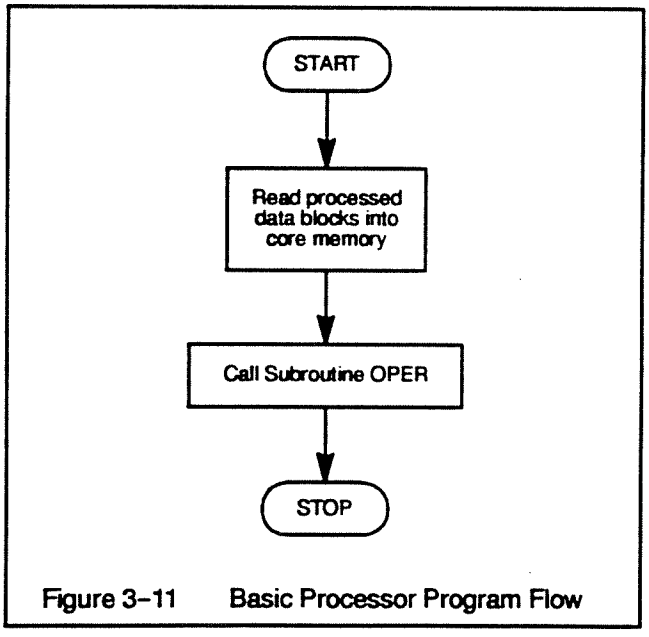


Figure 3–11 Basic Processor Program Flow

No general flow chart of the OPERATIONS DATA block can be presented here because everything in the block must be placed there by the user and the sequence of operations will always be specific to a particular problem. However, the flow chart for a typical OPERATIONS DATA block is shown, for illustrative purposes, in Figure 3–12. The last line in each block of this flow chart shows the SINDA operation which will accomplish the action described there in words. The various subroutines referenced by name (i.e., D1DEG1, FWDBCK and TPRINT) exist in prewritten, "canned" form in the SINDA/FLUINT library, and are described in Section 6.

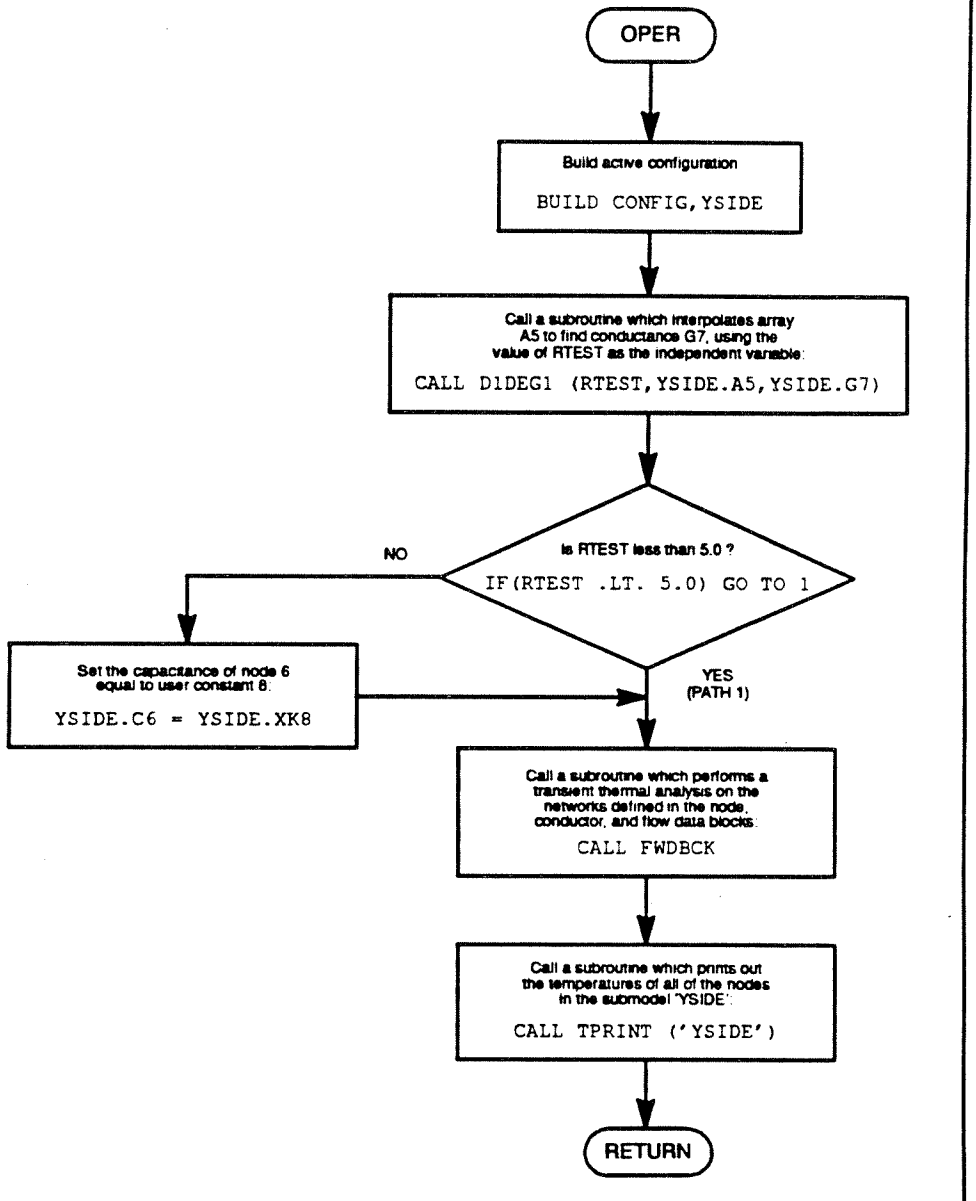


Figure 3–12 Sample Flow Chart for the Operations Data Block

It will be noticed that this flow chart still makes no explicit mention of subroutines VARBL0, VARBL1, VARBL2, FLOGI0, FLOGI1, FLOGI2, or OUTCAL. This will normally be true for any OPERATIONS DATA block flow chart because these subroutines are called automatically from within the prewritten network solution subroutines (e.g., FWDBCK); they are not, in general, called directly by the user. For example, OPER might call FWDBCK; FWDBCK, at certain points in the sequence of heat transfer and fluid flow calculations, calls VARBL0, VARBL1, VARBL2, FLOGI0, FLOGI1, FLOGI2, and OUTCAL for each active submodel; each of these routines in turn, again as directed by the user, calls upon various library subroutines to perform operations which are unique to the problem at hand. Thus, the operations included in the VARIABLES n, FLOGIC n, and OUTPUT CALLS blocks are used to customize the canned network solution routines so that they can accommodate the peculiarities that are specific to a given user's current problem. Figure 3–13 will aid the user in visualizing what has just been described—namely, that the operations entered by the user in the VARIABLES n, FLOGIC n, and OUTPUT CALLS blocks serves as "customized additives" to the prewritten operations contained in the thermal and fluid network solution subroutines.

In order to illustrate the way multiple logic blocks are incorporated into the program when the problem involves more than one submodel, a flow chart of subroutine VARBL0 for a multiple submodel problem is presented in Figure 3–14. For this illustration, several concepts involved in subroutine execution are introduced. In order to execute subroutine VARBL0, the following statement must appear in a "calling routine", e.g., FWDBCK:

```
CALL VARBL0 (NSM)
```

where NSM is the number of the current submodel (as found by the function MODTRN—or for fluid submodels, MFLTRN). This statement is known as the "calling sequence" for VARBL0. NSM is an *argument* of VARBL0 and may be a literal integer number or a variable name that points to an integer number in core memory.

In reference to Figure 3–15, note that a number of calls to subroutine VARBL0 are made, one for each active submodel. Each call to VARBL0 executes a specific VARIABLES 0 block (specified by the submodel name appearing on its header record) followed by calls to subroutines QVTIME and GVTIME, again for a specific submodel named in the calling sequences (see Figure 3–14). This example also shows that submodels are identified within SINDA/FLUINT by sequence numbers from 1 to n for an n-submodel problem.

Subroutines QVTIME and GVTIME perform the interpolations necessary to update time-dependent Q-source values (QVTIME) and conductor values (GVTIME) defined in the conductor and source data blocks using the time-dependency options (e.g., TVS, CYC, PER). If no such options were used, these calls will exist, but do nothing. The calls are placed at the end of the user's logic by the preprocessor, assuming that it is acceptable to perform these updates after the user's logic is executed. If this is not acceptable, the user may insert one or both of these calls at the beginning or anywhere within his VARIABLES 0 logic. This is done with specialized M-type statements:

```
CALL QVTIME('smn')
```

or

```
CALL GVTIME('smn')
```

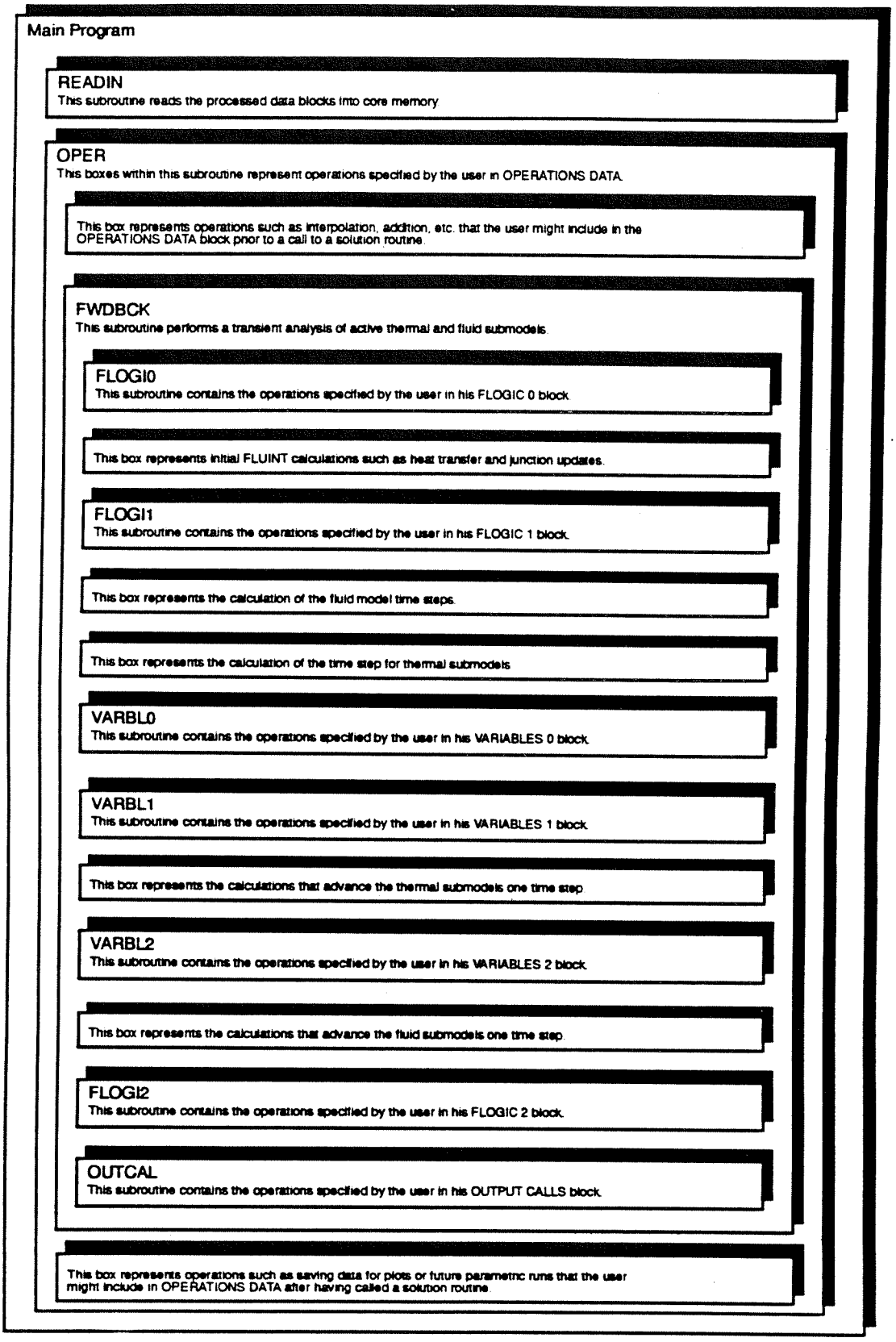


Figure 3–13 Nested Structure of the Logic Blocks

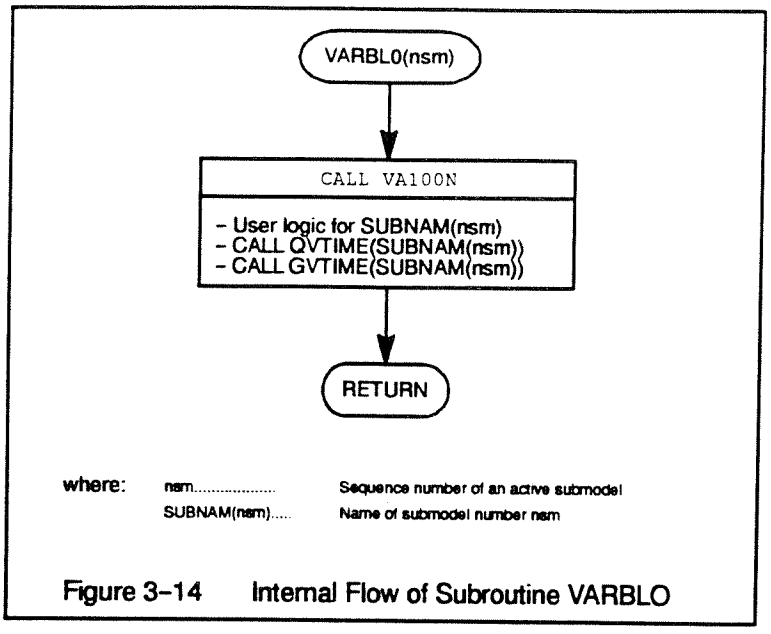


Figure 3–14 Internal Flow of Subroutine VARBLO

where smn must agree with the submodel name argument appearing on the preceding header record. When the preprocessor encounters such statements, it will refrain from inserting an identical call at the end of the user's logic.

With some specific differences, Figure 3–14 also represents the internal flow of subroutines VARBL1, VARBL2 and OUTCAL. The differences are that in VARBL1, QVTEMP, GVTEMP and CVTEMP replace the QVTIME and GVTIME calls. FLOGI0, FLOGI1, FLOGI2, VARBL2, and OUTCAL have no preprocessor-supplied calls.

The flow chart in Figure 3–15 details the sequence of operations contained in network solution routine, FWDBCK. (Figures 3–16 and 3–17 perform the same function for FORWRD and STDSTL/FASTIC.) Careful examination of this flow chart will reveal the extensive versatility which may be built into the solution routines by the creative use of the VARIABLES n, FLOGIC n, and OUTPUT CALLS blocks. The VARIABLES 0 blocks allow the user to interject operations which will be performed after the problem time has been incremented but before the actual heat transfer equations are integrated. The VARIABLES 1 block is used similar to VARIABLES 0, except that thermal time steps have been predicted at this point in the solution. In steady-state routines, the differences are greater—the VARIABLES 0 block is executed once per analysis, while the VARIABLES 1 block is executed once per iteration. The VARIABLES 2 block allows the user to interject operations which will be performed after the equations are integrated for the current time step, and the OUTPUT CALLS blocks allow the user to interject operations which will be performed only when the problem time has progressed to a multiple of some specified interval (i.e., the OUTPUT control constant).

The flow chart also reveals how the control constants can be used to achieve program flow control within the network solution routines. For example, the control constant OUTPUT (actually an array of control constants, one per submodel) is used to specify the time interval at which subroutine OUTCAL will be performed. Other control constants provide checks on the time step used, temperature change calculated per iteration, etc., as discussed in Section 4.6. Suffice it to say that by examining and modifying these constants in the VARIABLES n,

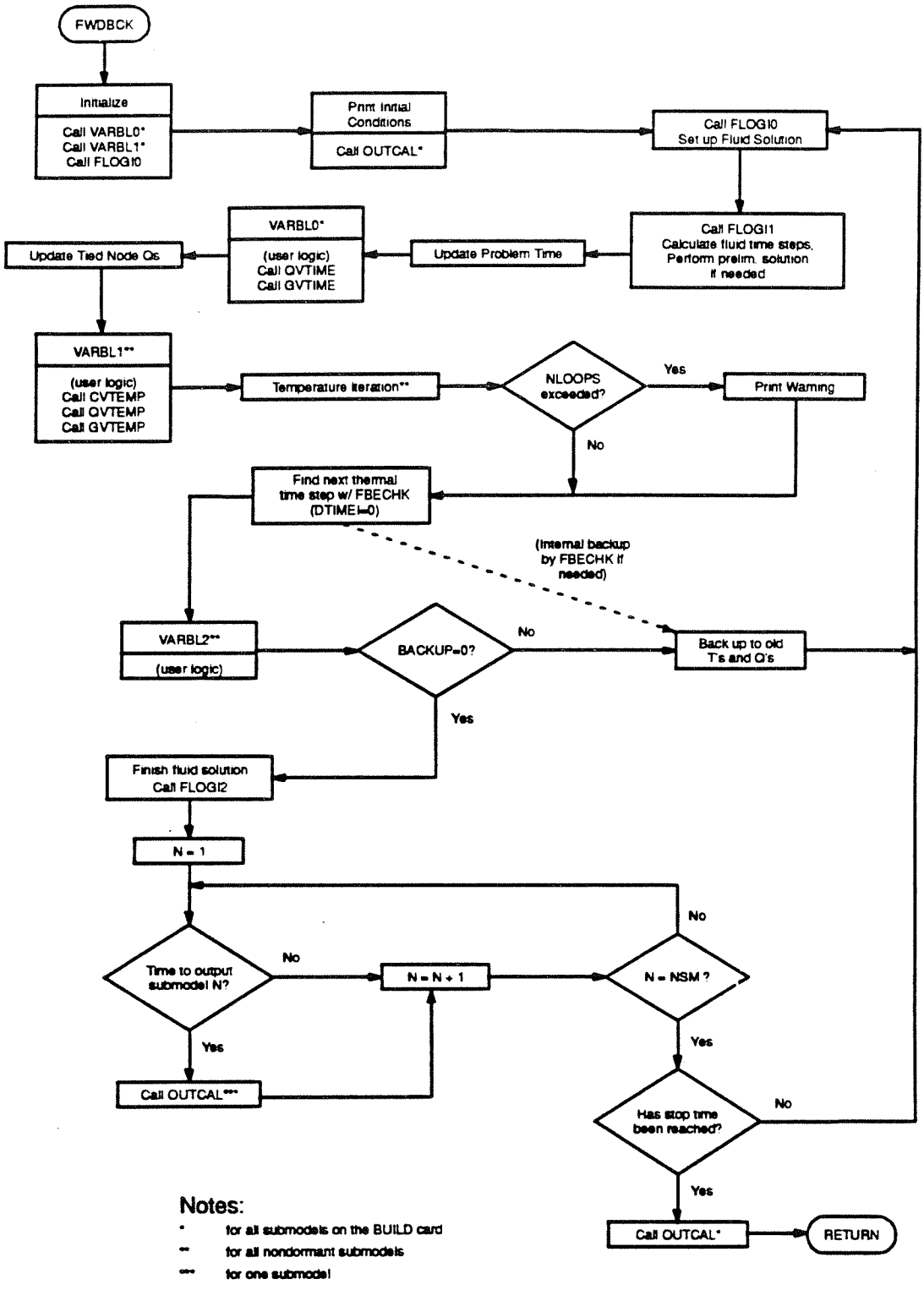


Figure 3–15 Flow Chart of Network Solution Routine FWDBCK

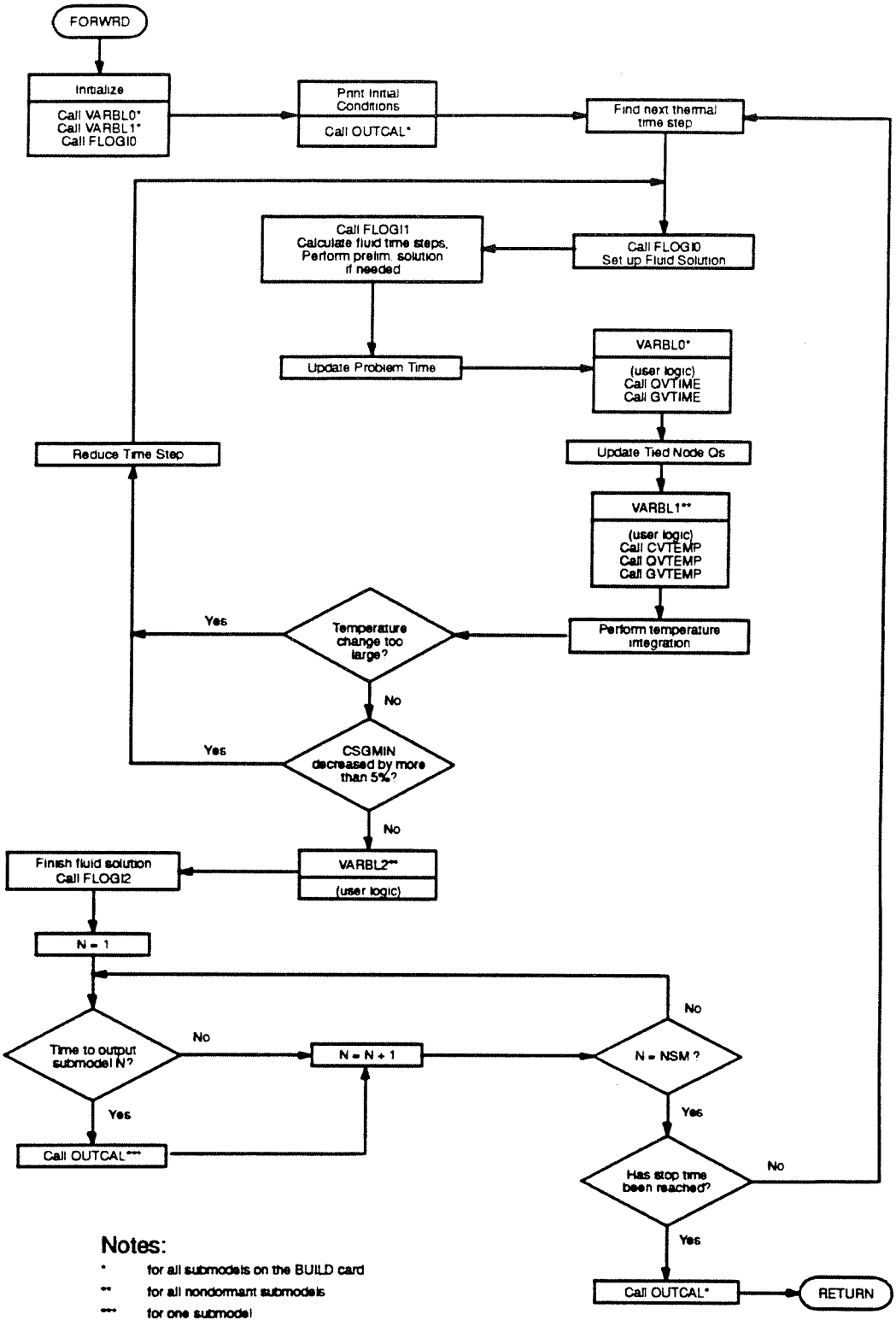
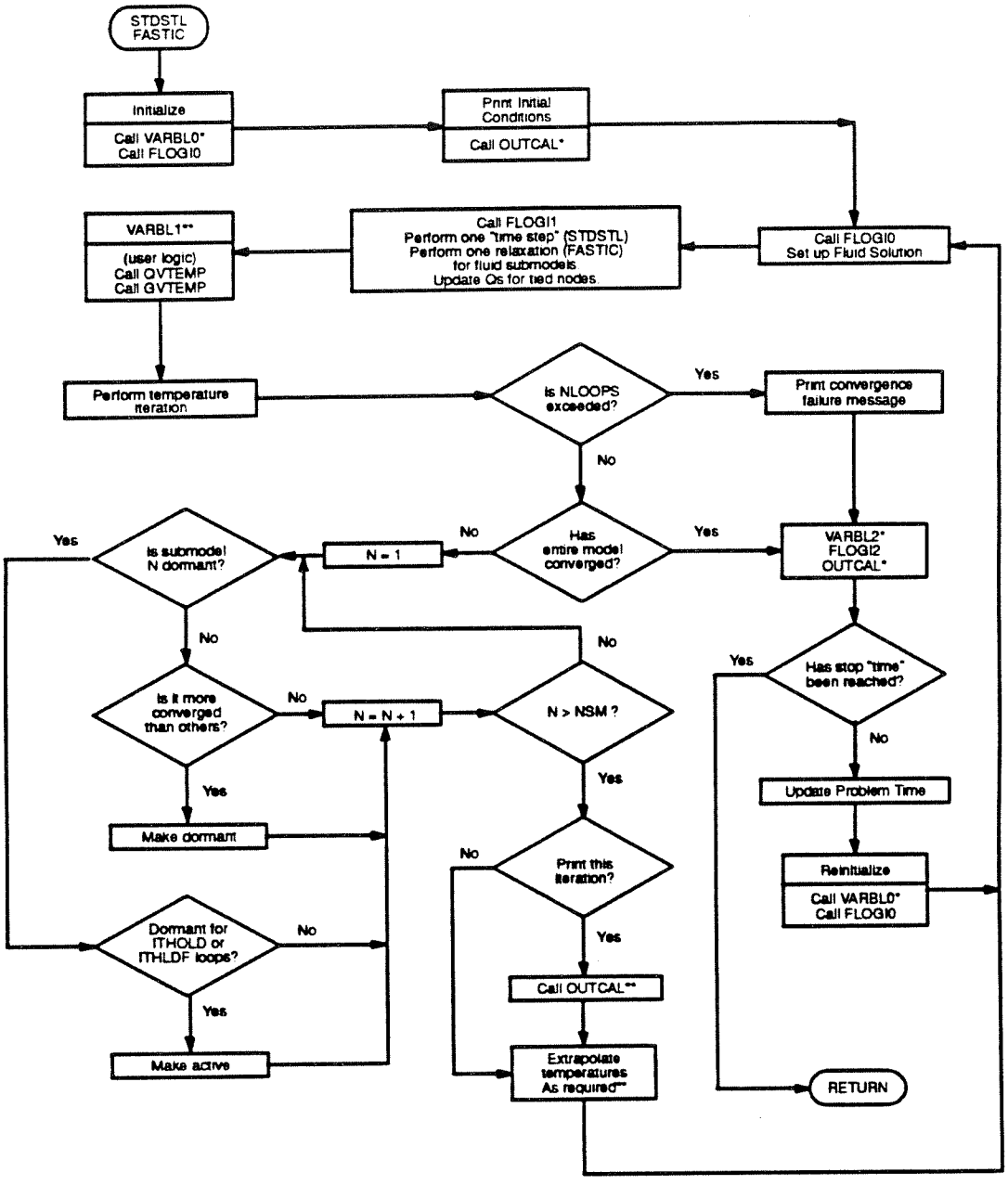


Figure 3–16 Flow Chart of Network Solution Routine FORWRD



Notes:

- * for all submodels on the BUILD card
- ** for all nondormant submodels

Figure 3-17 Flow Chart of Network Solution Routines STDSTL and FASTIC

FLOGIC n, and OUTPUT CALLS blocks, the user is able to effect complete control over the flow of the network solution routines, in addition to interjecting his own operations into the prewritten sequence of calculations contained therein.

The sample flow chart for the OPERATIONS DATA and VARIABLES 0 blocks (Figures 3–12 and 3–14) represent the result of applying three basic concepts (i.e., flow chart development, subroutine usage, and program flow control) to a specific problem. These concepts are equally applicable to the VARIABLES n, FLOGIC n, and OUTPUT CALLS blocks. The structure of the flow chart indicates the exact sequence of operations which the user has established as being necessary for the solution of his problem. The discrete operations detailed in each box are mechanized through the usage of prewritten subroutines from the library, and the Fortran "IF" statement (in the diamond) is used to accomplish program flow control (i.e., the selection of alternate sequences of operations). These three concepts will generally be applied by the user to each individual problem which he desires to solve. Various guidelines for structuring the flow of an OPERATIONS DATA block will be presented in Section 3.12.2, but the user must exercise a certain degree of judgment in applying them to his problem. In addition, the user should recognize that the SINDA/FLUINT system provides the capability and freedom to innovate in those cases where general guidelines cannot be applied. Similarly, the actions and generalized calling sequences of the subroutines in the SINDA/FLUINT library will be given in Section 6, but it is up to the user to select the routines which are appropriate for his problem, as well as to supply them with appropriate actual arguments. In the same fashion, Section 4 describes the available methods for achieving program flow control, but the responsibility for selecting and correctly applying these methods rests with the user.

3.12.2 Applications Guidelines

OPERATIONS DATA Block — When developing the OPERATIONS DATA block for a problem, it is convenient to partition the sequence of operations into two segments. One segment will consist of those operations performed prior to calling a network solution routine, and the other segment will consist of those operations performed after calling a solution routine. The first segment will contain everything necessary to set up and initialize the problem, and the latter segment will contain everything necessary to wrap up, print out, summarize, and/or save a problem which has been solved.

There is no reason, of course, why the OPERATIONS DATA cannot consist of a series of these initialize-solve-wrap up sequences, and parametric options have been provided to aid in such usage. Unless a specific action has been taken to recall earlier results, all operations act sequentially on the active networks such that the final conditions of the first operation are the initial conditions for subsequent solutions. *Recall that the first segment of an OPERATIONS DATA block must be a BUILD and/or and BUILDF instruction*, and that such instructions may be repeated as needed throughout the OPERATIONS DATA block. Any submodel not built will become inactive, and will return to its last status upon reactivization by another BUILD or BUILDF command.

The following operations are among those which would be appropriate for inclusion in the pre-solution segment:

1) Compute scale factors and other constant values which were inconvenient to compute by hand.

2) Scale data to consistent units.

3) Initialize required control constants if necessary.

4) Open any user files to be used during the run via the USRFIL routine.

In performing such set-up and initialization operations, the user should recognize that, while all memory locations initialized by the data blocks (i.e., temperature locations, conductance locations, source locations, flow data locations, etc.) are accessible in the OPERATIONS DATA blocks, not all of these locations contain meaningful values prior to calling a network solution routine. The following list describes the status of all accessible memory locations upon entering subroutine OPER:

1) All temperature locations contain the "initial temperature" of their corresponding node, as defined on the NODE DATA records.

2) Capacitance locations corresponding to nodes which were defined using the standard (3 blanks), CAL, or GEN options contain the capacitance values defined on the input records.

3) *Capacitance locations corresponding to nodes defined by the automated, variable capacitance options (SIV, DIV, etc.) contain no meaningful values.* These locations will be automatically loaded with the current capacitance of their respective nodes (as computed according to the options specified in the NODE DATA block) during each iteration of the heat transfer equations, and hence, they contain meaningful values only after a network solution routine has been started.

4) Source locations defined by the standard and GEN options will be initialized to the values on the SOURCE DATA records.

5) *Source locations defined by the temperature dependent and time dependent options (SIV, SIT, etc.) will contain meaningless data* for the reasons similar to those given in Item 3.

6) Conductance locations corresponding to conductors which were defined with the standard (3 blanks), CAL, or GEN options contain the conductance values specified on the input cards.

7) *Conductance locations corresponding to conductors defined by the automated variable conductance options (SIV, DIV, etc.), contain meaningless values* for reasons similar to those given in Item 3.

8) All user constants locations contain the values assigned in the USER DATA.

9) Control constants mentioned in the CONTROL DATA blocks contain the values assigned therein or defaulted by the preprocessor; all others contain zero.

10) All array locations contain the values assigned in the ARRAY DATA blocks (locations reserved with the SPACE option contain zeroes).

11) All fluid variable locations are filled with the exceptions of QDOT for tied lumps; QTIE for all ties and UA more some ties; GK, HK, EI, EJ, and DK for all connectors; AC, FC, and FPOW for all tubes and STUBE connectors, and FD, FG, and AM for those that are twinned.

While it is impossible to directly scale the value of a parameter which falls in the category of Items 3, 5 or 7 above, these parameters may be altered indirectly by modifying the value of their associated multiplying factors. This approach may be effected only if the multiplying factors are input as user constant references (instead of floating point data values).

An operational description of each of the network solution routines is given in Section 6 of this manual. Information on how to choose between solution routines is found in Section 4.

The following operations are among those which would be appropriate for inclusion in the post-solution segment of the OPERATIONS DATA block:

1) Perform summary calculations.

2) Print final values of interest in special formats.

3) Call for plots of data accumulated during the solution of the network.

4) Save on a permanent file any values which might be needed for later runs.

When a network solution routine has finished its calculations, it returns to the next operation in sequence in the OPERATIONS DATA block. At this point, all data locations (temperature, capacitances, control constants, etc.) contain the current values of their respective parameters, which may be used as initial conditions for subsequent solution routines.

VARIABLES 0 Block — The following operations are appropriate for inclusion in the VARIABLES 0 Block:

1) Update the current values of time-dependent quantities.

2) Examine the estimated time step about to be used and modify if desired.

3) Save current values of time-dependent boundary conditions and conductances.*

4) Compare current time-dependent boundary conditions and parameters with those of the previous time step.

5) Modify the time step and other control constants as desired for the impending time increment.

Upon entering VARBLO, DTIMEU (the time step used by the last pass through the temperature update algorithm) is available. Problem times TIMEN and TIMEM have also been updated, so the necessary initialization has been done for updating time-dependent variables and boundary conditions. If the user entered no VARIABLES 0 blocks, these updates will be performed by preprocessor supplied calls to subroutines QVTIME and GVTIME (see Figure 3–14). If desired, the user can insert logic statements before and/or after the calls to QVTIME and GVTIME. Thus it is possible to save time-dependent variables in user arrays at the beginning of VARBL0, then compare them with the same variables after the QVTIME & GVTIME calls. The user may also examine parameter vs. time arrays that are known to contain fast transients for the interval TIMEM minus TIMEO in order to be certain that significant changes in boundary conditions are not being lost due to an over-long time step. If this is happening, the user can change DTIMEU, TIMEN and TIMEM, then loop back to the beginning of VARBLO.

VARIABLES 1 Block — The following operations are among those which would be appropriate for inclusion in the VARIABLES 1 block:

1) Update the values of temperature varying quantities (i.e., capacitances, conductances, boundary temperatures, heat rates, etc.)

2) Modify control constants relevant to the network solution routine in progress.

Upon entering VARBL1, for the first iterative pass through the temperature solution algorithm, all variables will be just as they left VARBL0. VARBL1 is called before each iterative pass through the temperature solution loop in STDSTL and before each time step in FORWRD or FWDBCK but after the time step itself has been predicted. (If fluid submodels are active, it is impossible to precisely know a time step size before the actual solution—the available time step is therefore only an estimate.) The VARIABLES 0 and 1 blocks are thus the place for the user to enter any logic statements that use the temperatures that are correct at the end of the previous time step, as opposed to temperatures at the end of an iteration, which are merely intermediate values encountered in the iterative solution. The temperatures are accessed with the usual Tn or T(n+i) syntax. The user may access temperatures that are one time step old with the syntax TOLDn or TOLD(n+i).

Unless instructed otherwise, the preprocessor supplies calls to CVTEMP, QVTEMP and GVTEMP at the end of VARBL1. If there are heater nodes a call to HVACAL will also appear. These calls update the values of the capacitances, sources and conductances defined, in the pre-

* Beginning with SINDA '85, it is no longer necessary to save temperatures. The TOLD and T arrays contain temperatures for the previous and the current time steps, respectively.

ceding data blocks, using the automated variable capacitance source and conductance options. HVQCAL calculates heat flow to and from heater nodes. Unlike previous versions of SINDA, this is a default option, not the only option. If the user desires, he may insert his own calls to CVTEMP, QVTEMP or GVTEMP in one or more VARIABLES 1 data blocks. Using CVTEMP as an example, these calls are made as follows:

```
CALL CVTEMP ('smn')
```

where smn is the submodel name appearing on the HEADER VARIABLES 1 record. If the preprocessor encounters such calls in a VARIABLES 1 block, they will not be supplied at the end of the block. Making CVTEMP, QVTEMP and GVTEMP calls available to the user adds considerable flexibility to VARIABLES 1 operation. For example, suppose the user has the following entry in the NODE DATA block:

```
SIM 25,3,5,70.0,A9,XK5
```

This record generates nodes 25, 30 & 35, all with the same temperature-dependent capacitance. If the user desired to increase the value of one of these capacitances in VARBL1, the following logic would do this:

```
CALL CVTEMP ('smn')
C30 = 1.1*C30
```

Without this option the user could change the capacitances only indirectly, by changing XK5, which would affect C25 and C35 as well as C30. Prior to SINDA/FLUINT, the user would have had no choice but to enter the nodes individually and enter a separate user constant that applied to C30 only.

Since a location is provided in the Q source array for each diffusion and arithmetic node, there is nothing that forces the use of the SOURCE DATA block at all. Sources may be computed and entered into the Q-locations entirely within OPERATIONS DATA, VARIABLES 1 and/or VARIABLES 0 blocks.

Similarly, the user may find that there is no way to define a capacitance, source or conductance using anything other than special logic not provided in any of the regular input options. In this case, the special logic appears in a VARIABLES 0 or VARIABLES 1 block depending upon whether time or temperature is the appropriate independent variable. The user must keep in mind that the calls to QVTIME, GVTIME, CVTEMP, QVTEMP and GVTEMP must precede this logic if any of the automated data entry options were used for the variable to be defined. *Otherwise the results of this user-supplied logic will be overwritten at the end of VARBL0 or VARBL1.* It should be emphasized that a user-supplied call to an xVTIME or xVTEMP routine affects only a single submodel rather than all thermal submodels.

It can be seen by examining Figures 3–15, 3–16 and 3–17 that the relationship between VARBL0 and VARBL1 is different depending on the temperature solution routine. For explicit forward differencing (FORWRD), VARBL0 and VARBL1 could be combined into a single routine, which was the situation for all the solution routines in previous versions of SINDA. In

forward-backward differencing (FWDBCK), the only difference between VARBL0 and VARBL1 is that the time step prediction is current in VARBL1. The difference is greater for steady state solutions because VARBL0 is called once only, prior to beginning temperature iteration. This allows the user to initialize time-dependent program variables at some artificial TIME0 before beginning the steady state solution. VARBL1 is called once per iteration in steady state routines. If fluid submodels are active in any solution routine, then nodal source terms for tied nodes are updated between the VARBL0 and VARBL1 calls.

VARIABLES 2 Block — The following operations are appropriate for the VARIABLES 2 data blocks:

1) Examine changes in parameters that occurred during the last time step and set control constant BACKUP accordingly.

2) Accumulate net heat rates

3) Revise control constants

4) Predictor/corrector logic

5. Change-of-state calculations

6. Set new initial conditions for steady-state calculations

The following list describes the status of the thermal network parameters when subroutine VARBL2 is entered:

1. TEMPERATURES — Each location in the T-array contains the temperature of its corresponding node at the end of the current time step or steady-state solution.

2. CAPACITANCES — Each capacitance location contains the value of the capacitance used during the most recent iteration.

3. CONDUCTANCES — Each conductance location contains the value of the conductance used during the most recent iteration.

4. SOURCES — Each source location contains the value of the Q-source values used during the most recent iteration.

The flow charts for the transient network solution routines in Figures 3–15 and 3–16 indicate that the current iteration will be accepted as correct if the user does not set control constant BACKUP = 1.0 in VARBL2. Thus, the VARIABLES 2 blocks give the user the opportunity to examine the current iteration and to decide whether or not to proceed. This means that if BACKUP is set = 1.0 in any of the VARIABLES 2 blocks that correspond to submodels *not currently in the boundary state*, time will be set back for all of the submodels that participated in the last temperature update pass. Other parameters might also be adjusted in VARIABLES 2 according to some predictor/corrector scheme. Since VARBL2 is called only once for each time increment, the user can force each time step to be repeated, using the first pass for prediction and the second for correction. If fluid submodels are active, then they are not solved until all thermal back-ups have been completed, and the time step cannot be prescribed completely by DTIMEI.

If the cumulative heat flow in and out of certain nodes is pertinent to the problem, (e.g., for change of state calculations) VARBL2 is the appropriate place for this logic. The net change, over the most recent time step in the enthalpy of a diffusion node, can be obtained using subroutine DELH.

Program flow for steady state (Figure 3–17) is similar to the transient routines. VARBL2 is called only once after a steady-state solution has been found, rather than at each iteration approaching that solution. After calling VARBL2, TIMEN for each submodel is advanced DTIMES time units if that option is used. If the problem end time, TIMEND, has not been reached, a new steady-state solution is sought. Hence, the VARIABLES 2 data blocks may be used to compute the initial conditions applicable to the next solution. The user must remember, however, that the next solution will begin with a call to VARBL0, which may contain inappropriate initialization statements. These may be bypassed, on a submodel basis if desired, in VARBL0 by appropriate tests on the values of TIMEN, NSOL, or LOOPCT.

FLOGIC Blocks (General) — FLOGIC 0, FLOGIC 1, and FLOGIC 2 blocks are used similarly to VARIABLES n blocks for the flow model solution. The FLOGIC 0/1/2 sequence represents opportunities to respectively initialize/tailor/wrap-up the fluid solution. While both thermal and fluid variables are available in both FLOGIC and VARIABLES blocks, the user should only alter fluid variables in FLOGIC blocks and only thermal variables in VARIABLES blocks. Furthermore, some FLUINT variables (PL, TL, XL, HL, DL, VOL, and FR) should not be changed directly in these blocks. (See utility routines CHGLMP and CHGVOL for controlled changes in lump states.) Note that, unlike VARIABLES n blocks, there are no default operations in FLOGIC n blocks. Figure 4–6 illustrates the appropriate block in which to update, inspect, or modify tube and STUBE parameters (e.g., AC, TLEN, etc.).

FLOGIC 0 Block — This logic block represents an opportunity to initialize a FLUINT solution before a subsequent solution step. FLOGI0 is called at the beginning of each iteration in STDSTL or FASTIC (unlike its analog VARBL0), and at the beginning of each new time step solution in FORWRD and FWDBCK. Appropriate operations in this block can include:

1) Preparing for the heat transfer, pressure drop, and component model calculations.

2) Performing any transient control logic, such as updating valve or loss elements.

3) Make any calls to CHGLMP, HLDLMP/RELLMP, CHGSUC, PUTTIE, etc.

Upon entering FLOGI0, lump states and path flowrates have either been initialized or exist at the state resulting from a previous run or time step solution. Time steps for the fluid solution have not been calculated. Values that are appropriate to change at this point include QDOT, VDOT, COMP, CX, CY, CZ, IPDC, WRF, UPF, AC, FC, FPOW, HC, AM, FG, FD, FK, AF, AFT, DH, TLEN, TPF, SMFR, SVFR, SPD, RC, CFC, XVH, XVL, GK, HK, EI, EJ, DK, DUPI, DUPJ, UA, DUPN, DUPL, and any control constants.

FLOGIC 1 Block — This logic block represents an opportunity for advanced users to tailor a FLUINT solution after most calculations have been made in preparation for a solution step, but before a solution step has actually been taken. FLOGI1 is called at least once each iteration for steady-state solutions and at least once each time step for transient solutions. Appropriate operations in this block can include preparing or tailoring any heat transfer, pressure drop, and device and component model calculations. Note that *this is **not** the correct place to make major changes* or to call PUTTIE, CHGVOL, CHGLMP, HLDLMP, or RELLMP. Unlike VARIABLES 1, *FLOGIC 1 should not be used as the principal logic block.*

Upon entering FLOGI1, all operations prior to the time step prediction and the subsequent solution have been performed. Changes to heat transfer ties and connector device and component model parameters will be ignored until the next solution step. Changes to lump states or

similar changes can cause errors. However, except for FASTIC runs this is the appropriate place to change tank QDOT, COMP, and VDOT values, connector GK, HK, EI, EJ, and DK values (which have now been calculated for all connectors), and any tube parameters other than those relating to friction options. Time steps for the fluid solution have not yet been calculated, and cannot be completely known before a solution has occurred.

FLOGIC 2 Block — This block represents an opportunity to wrap-up the last solution step and/or prepare for the next. It can be used similarly to FLOGIC 0, and is called once after each steady-state solution and once after each time step has been taken.

OUTPUT CALLS Block — All submodels for which data is output to a printer or permanent file (other than the restart file) require an OUTPUT CALLS data block. It is therefore appropriate but not required that there be an OUTPUT CALLS data block for each thermal or fluid submodel.

The preprocessor converts the OUTPUT CALLS logic statements as required and consolidates them into a single subroutine OUTCAL in the manner similar to that shown in Figure 3–14. One notable difference between OUTCAL and the VARBLn blocks is that the preprocessor does not insert subroutine calls into the Fortran logic, thus, *there is no default output.*

The portion of OUTCAL that applies to a given submodel is executed whenever the value of the current problem time or iteration count advances to a multiple of the output interval for that submodel. The output intervals are defined by an array of values of control constant OUTPUT and ITEROT, one value per thermal submodel, and OUTPTF and ITROTF, one value per fluid submodel. OUTCAL may be called after every solution step. This is done for individual thermal submodels in transients by setting elements in the control constants array OPEITR to 1 (use OPITRF for fluid submodels); setting ITEROT and/or ITROTF to 1 forces OUTCAL each steady-state iteration.

OUTCAL is called at the beginning of each solution routine to record initial conditions. It will also be called in the event of an abort due to a condition found by the program.

The following are appropriate operations to include in the OUTPUT CALLS blocks:

1) Print values of interest

2) Save current values on a permanent file for postprocessors and graphics programs.

3) Save current values on a program file for printing or plotting in the post solution segment of the OPERATIONS DATA block.

4) Write to a restart output file or to a crash file.

5) Write current values to a program file for initialization of solutions later in the run.

These operations are all quite straight forward and are implemented using several library subroutines that offer a variety of printout formats and file writing capabilities. Some of these routines allow the user to supply multi-word character strings as column header titles and parameter labels. The CARRAY DATA blocks provide a convenient means of entering this data. The PAGHED routine may be used to change the presentation of such output routines.

All the program variables and parameters are available for use in OUTPUT CALLS logic, as is also true in the VARIABLES n and FLOGIC n blocks. This allows the user to control output operations based on the values of any number of program variables.

SUBROUTINE DATA Block — The SUBROUTINE DATA block is intended to contain any subroutines the user wishes to call from the other logic blocks. Like the OPERATIONS DATA block, there is just one SUBROUTINE DATA block, though it may contain any number of user-supplied subroutines. The subroutines may utilize F or M-type statements which means that temperatures, conductors, pressures, arrays and the like may be referenced using the smn.Xn or smn.Z(m+i) forms. Note that, as in the OPERATIONS DATA block, the smn prefix must always be supplied (see also the DEFMOD instruction).

CALL COMMON is a macroinstruction unique to SUBROUTINE DATA. It includes the labeled common blocks that are automatically supplied at the beginning of the other logic blocks (see below), allowing access to all of the same SINDA/FLUINT variables. This record is optional since it is not always necessary for a particular subroutine to reference the program variables. When used it must immediately follow the subroutine declaration.

Thus, the user has a choice of whether any routine in SUBROUTINE DATA should be (1) treated like other logic blocks, e.g., have access to central variables like C, G, DH, PL, and QTIE and control constants like TIMEND, OUTPTF, and NLOOPS, or (2) whether the routine should be more isolated, with all variables considered local, thereby alleviating concerns regarding collisions with reserved SINDA or FLUINT words. In the latter case, the FSTART/FSTOP and DEBOFF/DEBON instructions are very useful to turn off translation and debug, respectively. (No debug checks are performed if translation is off, but the reverse is not true: translations can still occur if the debug feature is turned off.)

The generic format of the SUBROUTINE DATA block is as follows:

```
HEADER SUBROUTINE DATA
        SUBROUTINE SUBR1 (arg1, arg2,--argn)
        CALL COMMON
        .          .          .
        .          .          .
        .          .          .
        M and/or F-type statements
        .          .          .
        .          .          .
        .          .          .
        END
C
FSTART
        SUBROUTINE SUBR2 (arg1, arg2, arg3,---argn)
                .          .          .
        .          .          .
        .          .          .
        F-type statements
        .          .          .
        .          .          .
        .          .          .
        END
FSTOP
                .
                .
                .
```

As in any Fortran program, the argument list is optional. The END record at the conclusion of each subroutine is mandatory. The first subroutine SUBR1 is an example of a routine which will be treated like any other logic block. The second routine, SUBR2, lacks the CALL COMMON command and therefore any access to central variables and control constants. Translation is therefore turned off to avoid the need to worry about name collisions—any valid Fortran variable name may now be used in SUBR2 such as "T", which will no longer mean a nodal temperature.

Program Common — The CALL COMMON macro-instruction supplies labeled common blocks, which are always present in any FLOGIC, VARIABLES, OUTPUT CALLS, or OPERATIONS DATA block, *and which constitute reserved variable names*. The following pages detail exactly what is supplied in most logic blocks, and in SUBROUTINE DATA blocks into which CALL COMMON has been inserted.

Common Name	Variable Name	Description
CNTRL1	(below)	Arrays of control variables (one per submodel)
CNTRL2	(below)	Same as CNTRL1
NUMCON	K	Numbered user constants
USERC	(many)	Named user constants
ARAYS	A	User arrays (other than T-arrays)
CARAYS	UCA	Character arrays
TAPES	(many)	Names of all program files
NDAT1	T	Temperature (nodal) array
NDAT2	C	Capacitance array
NDAT3	Q	Source (nodal) array
NDAT4	NMOD	Submodel name array
NDAT5	NSTRT	Array of pointers to each submodel in the PCS
NDAT6	NMDIF	Number of diffusion modes in each submodel
NDAT7	NMARI	Number of arithmetic nodes in each submodel
NDAT8	NMHT	Number of heater nodes in each submodel
NDAT9	NMBD	Number of boundary nodes in each submodel
NDAT10	NDNAM	Array of node names
NDAT11	NDINT	Array of actual node numbers
NDAT12	TOLD	Old Temperature Array
CDAT1	PT	Pointers to nodes
CDAT2	PG	Pointers to G values
CDAT3	NLIN	Number of linear conductors connected to a node
CDAT4	NRAD	Number of radiation conductors connected to a node
CDAT6	CNAM	Array of model names
CDAT7	NCTOT	Total G per model
CDAT8	NGSTRT	starting G per model
CDAT9	G	Array of Conductor values
LUMP4	HL	Enthalpy array
LUMP5	PL	Pressure array (DOUBLE PRECISION)
LUMP6	XL	Quality array
LUMP7	TL	Temperature (lump) array
LUMP7A	DL	Density array
LUMP10	VOL	Tank volume array
LUMP11	VDOT	Tank volume rate of change array
LUMP12	QDOT	Source (lump) array
LUMP30	COMP	Tank compliance factor
LUMP31	CX	Lump X coordinate
LUMP32	CY	Lump Y coordinate
LUMP33	CZ	Lump Z coordinate
TIEDAF	QTIE	Tie heat rate (before duplication factors applied)
TIEDAN	DUPL	Tie duplication factor, lump side
TIEDAO	DUPN	Tie duplication factor, node side
TIEDA7	UA	Tie conductance (before duplication factors applied)
PATH3	FR	Flowrate array
PATH4	HC	Tube (STUBE) Head coef. array*
PATH5	FC	Tube (STUBE) Friction coef. array*
PATH6	AC	Tube (STUBE) Area change coef. array*
PATH7	GK	Connector Pressure/flowrate coef.
PATH10	HK	Connector Flowrate offset
PATH13	DH	Tube (STUBE) Diameter array*
PATH14	TLEN	Tube (STUBE) Length array*
PATH15	AF	Tube (STUBE) Flow area array*
PATH16	WRF	Tube (STUBE) Wall roughness array*
PATH17	UPF	Tube (STUBE) Upstream fraction array*
PATH18	IPDC	Tube (STUBE) Pressure drop correlation array*
PATH23	DUPI	Path upstream duplication factor
PATH24	DUPJ	Path downstream duplication factor
PATH26	FPOW	Tube (STUBE) Flowrate exponent in FC term*
PATH30	EI	Path deriv. of flowrate wrt defined upstream enthalpy
PATH31	EJ	Path deriv. of flowrate wrt defined upstream enthalpy
PATH32	DK	Path deriv. of flowrate wrt twin's flowrate (twins)
PATH33	FD	Tube (STUBE) Interface drag factor (twins)*
PATH34	FG	Tube (STUBE) Phase generation factor (twins)*
PATH35	AM	Tube added mass coefficient (twins)
PATH40	AFT	Tube (STUBE) and LOSS family throat area*
PATH41	FK	Tube (STUBE) and LOSS family K-factor*

* Device and component model data (such as RC, CFC, SMFR, SVFR, etc. may also be stored in these arrays. The user need not worry about this fact *as long as he doesn't treat them as scratch space* because the all valid references to these data (such as LOOP.RC10) are translated into the correct storage cell by the preprocessor, as can be seen in the Fortran list file.

Common Name	Variable Name	Description
FSIZ1	(below)	Total fluid model sizes
FSIZ2	NUMTNK	Number of tanks in each submodel
FSIZ3	NUMJNC	Number of junctions in each submodel
FSIZ4	NUMPLE	Number of plena in each submodel
FSIZ5	NUMTUB	Number of tubes in each submodel
FSIZ6	NUMCNT	Number of connectors in each submodel
FSIZ7	NTOTL	Number of lumps in all submodels
FSIZ8	NTOTK	Number of paths in all submodels
FSIZ11	FLID	Fluid sequence number for each submodel (INTEGER)
SOLN25	DTIMEF	Time step required by each submodel
FLUCNT	(below)	Fluid submodel control constants
FLUDAT	FI	Fluid properties array
	FLDPNT	Pointer into FI for each fluid type (INTEGER)
	LIST	List of valid fluid IDs corresponding to FLDPNT entries
FLUCN1	UID	Unit system flag (INTEGER)
	PATMOS	Global control constant (abs. zero in pressure units)
	ACCELX,Y,Z	Acceleration vector components
SOLTYP	NSOL	Solution routine identifier

Dimensions:

NDAT1, NDAT2, NDAT3, NDAT10, NDAT11 and NDAT12 are dimensioned NNOD.
NDAT4 through NDAT9 are dimensioned MMODS.
CDAT3 is dimensioned NNOD
CDAT1 is dimensioned NGT**2 + NMBD
CDAT4 is dimensioned NNOD
CDAT2 same as CDAT1
CDAT5 is dimensioned NNOD
CDAT6 is dimensioned NGT
CDAT7 is dimensioned MMODT
CDAT8 is dimensioned MMODS
CDAT9 is dimensioned (total number of conductors) NGT
TIEDAF through TIEDA7 are dimensioned MXTIE
LUMP4 through LUMP7A and LUMP 12 are dimensioned NTL
LUMP10 and LUMP11 are dimensioned NTT
PATH3 through PATH41 are dimensioned NTK; PATH35 is dimensioned to NTTB
FSIZ2 through FSIZ11 and SOLN25 are dimensioned MXMODF (max. no. of fluid submodels)

Certain parameters are initialized, describing the dimensions of most program arrays. These are:

Parameters:

MXMODS ..... Maximum number of thermal submodels
MXMODF ..... Maximum number of fluid submodels
MXNODE ..... Maximum number of nodes
MXCOND ..... Maximum number of conductors
MXUARY ..... Maximum number of arrays
MXTARY ..... Maximum number of TARRAYS
MXTOKN ..... Maximum number of tokens per input line
MXCNTR ..... Maximum number of thermal control constants
MXUCON ..... Maximum number of user constants (numbered)
MXNCON ..... Maximum number of user constants (named)
MXFRES ..... Maximum number of Fortran reserved words
MXFILE ..... Maximum number of files
MXFLID ..... Maximum number of fluid types (includes 20 library)

Except for MXMODS and MXMODF, *these parameters are not currently used,* and contain values that are appropriate for the preprocessor limits (and are, in fact, the values contained in the PARAMETER.INC file), but which do not reflect the actual array sizes. These values are reserved for future program versions. Use instead the values contained in CCONST and FSIZ1 (described below).

The contents of commons CNTRL1, CNTRL2, FLUCNT, and FLUDAT are arranged as follows. One variable per submodel is defined in memory. See Section 4.6 for the variable name definitions. Note the MXMODS submodel limit for thermal submodel, and the MXMODF submodel limit for fluid submodels, currently 100 and 25, respectively.

```
      COMMON/CNTRL1 /
     +        DTIMEU(MXMODS), DTIMEI(MXMODS), DTIMEL(MXMODS)
     +      , DTIMEH(MXMODS), OUTPUT(MXMODS), DTMPCC(MXMODS)
     +      , DTMPCA(MXMODS), NDTMPN(MXMODS), ITHOLD(MXMODS)
     +      , ATMPCA(MXMODS), NATMPN(MXMODS), DRLXCA(MXMODS)
     +      , NDRLXN(MXMODS), ARLXCA(MXMODS), ARLXCC(MXMODS)
     +      , NARLXN(MXMODS), CSGMIN(MXMODS), CSGFAC(MXMODS)
     +      , CSGMAX(MXMODS), NCSGMN(MXMODS), NCGMAN(MXMODS)
     +      , EBALSA(MXMODS), EBALNA(MXMODS), EBALSC(MXMODS)
     +      , EBALNC(MXMODS), NEBALN(MXMODS), NLOOPT(MXMODS)
     +      , EXTLIM(MXMODS), ITERXT(MXMODS), ITEROT(MXMODS)
     +      , BACKUP(MXMODS), OPEITR(MXMODS), DRLXCC(MXMODS)
     +      , ATMPCC(MXMODS), ESUMIS(MXMODS), ESUMOS(MXMODS)
      INTEGER OPEITR
C
      COMMON/CNTRL2/       NDTMPC(MXMODS), NATMPC(MXMODS)
     +      , NDRLXC(MXMODS), NARLXC(MXMODS), NCSGMC(MXMODS)
     +      , NCGMAC(MXMODS), NEBALC(MXMODS)
      CHARACTER*8          NDTMPC,       NATMPC
     +      , NDRLXC,       NARLXC,       NCSGMC
     +      , NCGMAC,       NEBALC
      COMMON/FLUCNT/
     +        DTSIZF(MXMODF),   DTMAXF(MXMODF),   OUTPTF(MXMODF)
     +      , RSSIZF(MXMODF),   RSMAXF(MXMODF),   RERRF(MXMODF)
     +      , REBALF(MXMODF),   ITROTF(MXMODF),   OPITRF(MXMODF)
     +      , DTMINF(MXMODF),   ITHLDF(MXMODF)
     +      , DTTUBF(MXMODF),   RSTUBF(MXMODF)
      COMMON/FLUCN1/ UID, PATMOS, ACCELX, ACCELY, ACCELZ
      DOUBLE PRECISION PATMOS
      INTEGER UID,OPITRF
      COMMON/FLUDAT/  FI(90), FDATA(1700)
```

Common CCONST contains the single-valued program variables. Their arrangement and definitions are as follows:

MMODS ..... Total number of thermal submodels
NNOD ..... Total number of thermal nodes
ABSZRO .... Absolute zero
SIGMA ..... Stefan - Boltzmann constant
NMACT ..... Number of active thermal submodels
NFACT ..... Number of active fluid submodels
DATE ..... Current date
TIMDY ..... Time of day
LINECT ..... Current line number on a printout page
PAGECT .... Current page number
DTIMES ..... Pseudo time step for steady-state
TIMEN ..... Current problem time
TIMEO ..... Old problems time (Previous TIMEN)
TIMEM ..... Average of TIMEN and TIMEO
TIMEND ..... Stop time
NLOOPS .... Number of steady state iterations allowed
LOOPCT .... Current iteration count
MLINE ..... Number of lines allowed on a page
NTOLD ..... Used to access TOLD array

Common FSIZ1 contains the single-valued program variables. Their arrangement and definitions are as follows:

NFM ..... Total number of fluid submodels
NTL ..... Total number of lumps
NTT ..... Total number of tanks
NTJ ..... Total number of junctions
NTPL ..... Total number of plena
NTK ..... Total number of paths
NTTB ..... Total number of tubes
NTC ..... Total number of connectors

Common SOLTYP contains one single-valued program variable: NSOL. This very useful variable identifies the current solution routine, which can be used to alter the instructions in user logic blocks (VARIABLES and FLOGIC) and in user subroutines:

NSOL =0 ..... FASTIC
=1 ..... STDSTL
=2 ..... FWDBCK
=3 ..... FORWRD
=4 ..... FWDMTS

Note that logic separated on the basis of steady-state vs. transient can use an NSOL comparison such as "IF(NSOL .LE. 1)" to mean "if called from a steady-state routine ..."

Summary of Translatable Parameters — All above parameters constitute the reserved word list, meaning that none of these names can be reused by users in their logic without causing name collisions. While all are available, only a subset are recognized by the preprocessor as translatable (i.e., can accept submodel prefixes and/or user ID suffixes, which are converted into global array references). Single-value numbers like LOOPCT, ABSZRO, ATEST through ZTEST and NSOL need no translation. The fluid identifier, FI, is translatable but follows unique rules (see Section 6.9.2). All control constants are translatable, given submodel prefixes. The remaining translatable variables, of the form [smn.]Zn, are:

Nodes	All	T, TOLD, Q (Q ignored if boundary)
	Diffusion	C
Conductors	All	G
Lumps	All	PL (double precision), TL, HL, DL, XL, CX, CY, CZ, QDOT (QDOT ignored if plenum)
	Tank	VOL, VDOT, COMP
Paths	All	FR, DUPI, DUPJ, GK, HK, EI, EJ, DK
	Tube/STUBE	HC, FC, FPOW, AC, IPDC, UPF, TLEN, DH, AF, AFT, FK, WRF, FD, FG
	Tube	AM
	MFRSET	SMFR (stored internally* in the HC array)
	VFRSET	SVFR (stored internally* in the HC array)
	LOSS/valves	FK, AF, AFT, TPF (includes CHKVLV and CTLVLV connectors; TPF is stored internally* in the WRF array)
	LOSS2	FKB (stored internally* in the AC array)
	CAPIL	RC, CFC, XVH, XVL (same for 1st CAPPMP connector; stored internally* in the DH, AF, TLEN, and WRF arrays, respectively)
	VPUMP	SPD (stored internally* in the FC array)
Ties	All	UA, QTIE, DUPL, DUPN
User data	Arrays	A or NA (equivalenced)
	Constants	XK or K (equivalenced)
	CARRAYs	UCA

* "Internally stored as ..." parenthetical comments: Actually, there are no real arrays named after most connector device mnemonics such as "RC" or "SMFR," except for STUBE parameters, all of which are real arrays that also contain tube data. To reduce memory requirements, device data for all other connector devices is stored within the arrays set aside for tubes and STUBEs. As an example, the SMFR for an MFRSET connector is stored internally within the program in the appropriate cell of the HC array, which is otherwise meaningless for that connector. The preprocessor converts the occurrence of "SMFR(" in logic blocks to "HC(" as can be seen in the output Fortran files. This has little impact on the user, except the strong caution that **none of these arrays should be used as scratch space.** Furthermore, many other data are stored in these arrays that are not translatable.

3.12.3 Operational Details

The following discussion presents the basic information required to prepare the logic blocks. Examples of properly prepared records are presented and will serve to introduce the user, by illustration, to the techniques of executing the activity described in flow charts. FLUINT operations will not be described explicitly here because they follow the same rules as the thermal operations.

The user is reminded that all logic appears beneath one or more of the following header records which may appear in any order. The OPERATIONS DATA block is the only one that *must* appear, although a run without OUTPUT CALLS would be a very special case.

```
HEADER OPERATIONS DATA
C         Executive (Driver) Operations
HEADER VARIABLES 0, smn
C         VARIABLES 0 Operations
HEADER VARIABLES 1, smn
C         VARIABLES 1 Operations
HEADER VARIABLES 2, smn
C         VARIABLES 2 Operations
HEADER FLOGIC 0, smn
C         FLOGIC 0 Operations
HEADER FLOGIC 1, smn
C         FLOGIC 1 Operations
HEADER FLOGIC 2, smn
C         FLOGIC 2 Operations
HEADER OUTPUT CALLS, smn
C         OUTPUT CALLS Operations
HEADER SUBROUTINE DATA
C         User Subroutines
```

The header records must appear in the format shown. *Blanks and commas serve to delineate key words and must appear.* More than one blank between key words is of no consequence, but consecutive commas must not appear.

As a further reminder, all logic consists of F-type Fortran statements and M-type Fortran statements. An F-type statement is any valid Fortran statement. An M-type statement is similar to an F-type statement, except that the *actual* numbering system may be used to reference data. That is, F-type statements are FORTRAN statements which are not translated by the preprocessor, whereas M-type statements are Fortran statements that are modified by the preprocessor to reflect translation from the *actual* to the *relative* numbering system. Another feature of M-type statements is the submodel name qualifier on certain variable names. This is another aspect of the translation from *actual* identification to the *relative* numbering system. The default statement is type M. That is, a blank in column 1 will be understood as an M-type statement. An F *must* appear to identify F-type statements. A block of F statements may also be

delineated by a FSTART and FSTOP. An integer number in Column 1 is also interpreted as an M. Thus, statement numbers may appear anywhere in columns 1 through 5 on M-type statements.

3.12.3.1 M-Type Statements

M-type statements may be used for any purpose that a regular Fortran statement can accomplish in a Fortran program. Thus, M-type statements may be thought of as a SINDA-specialized extension of the Fortran language. Nonstandard statements run the risk of generating compiler errors.

The following paragraphs illustrate, by example, the various formats and uses of M-type statements. In the examples, the M will appear for emphasis. Note, however, that *a blank or integer number in column 1 is equivalent to an M.* Also note that although the examples used in the next two sections are for thermal variables, almost all FLUINT variables are likewise accessible, as was summarized in the last subsection.

Subroutine Calls — M-type statements may be used to call subroutines. Such subroutines may reside in the SINDA/FLUINT library, the computer system library or as user-supplied subroutines in the SUBROUTINE DATA block. The subroutines in the libraries are precompiled Fortran subroutines. The subroutines in the SUBROUTINE DATA block are interpreted and compiled along with the other logic and thus may consist of M-type statements.

The general format for a subroutine call that requires no arguments is as follows:

```
120     CALL STDSTL                          $ EXAMPLE 1
M       CALL FWDBCK                          $ EXAMPLE 2
```

The number 120 in Example 1 is a statement number. Its only purpose is to provide a target point for program flow control using Fortran IF, GO TO, or DO statements (see Section 4.2). When no such target point is needed, the statement number field may be left blank. When a statement number is used, it should be unique to its data block, but it need not be chosen in any particular sequential order.

Example 1 shows an M-type statement which calls subroutine STDSTL. A statement number, 120, is used so that some such Fortran statement as GO TO 120 will cause program flow to proceed directly to this statement. Example 2 shows a statement that calls subroutine FWDBCK*. Note that both are understood as M-type statements, even though it wasn't explicit in example 1. These examples also show the use of the optional comment data field to the right of a $ delimiter. Comment data to the right of column 72 will be printed, unlike previous versions of SINDA.

M-type statements that call subroutines that require arguments are formatted similarly, as follows:

```
      CALL D1DEG1 (RTEST, A5, G7)            $ EXAMPLE 3
      CALL QVTIME ('SUBM2')                  $ EXAMPLE 4
```

* This routine will be introduced along with other solution routines in this section. Detailed descriptions of these routines may be found in Section 6.

In example 3, RTEST, A5 & G7 are arguments. A5 is a reference to User Array 5 and G7 is a reference to Conductor 7. RTEST is a user constant that should appear in USER DATA. Its appearance in USER DATA is not mandatory, however. The ramifications of this are explained below in Section 3.12.3.4.

Example 3 would be correct only if it appeared under a header record with the same submodel name that A5 and G7 were entered under. If Example 3 were under another submodel name, or if it were in the OPERATIONS DATA block or the SUBROUTINE DATA block, it would have to appear as follows:

```
CALL D1DEG1 (RTEST, SUBM1.A5, SUBM1.G7) $ EX. 3A
```

where SUBM1 is the submodel name associated with A5 and G7.

Both M-type and F-type statements must end prior to column 72. If this does not provide sufficient space, they may both be continued, according to the Fortran rule, if a continue character is placed in column 6 of the statements that follow, as shown below:

```
Column 6
     ||
101   CALL ADD (SUB1.T4, SUB2.T7, 9, SUB1.T21,
     +T36, SUB1.T3, SUB2.T8, T27, T19,       $ COMMENTS
     +T14, SUB2.T106, T9241, 308.4, XK5,
     +XK5, TTEST)                            $ EX. 5
```

It should be noted that:

1) Any statement numbers must appear only on the first line of multi-line statements.

2) Permissible arguments allowed on M-type subroutine calls include literal data values, user constants names, and references to program variables by *actual* identification numbers, with or without submodel qualifiers.

3) Arguments must be separated by commas. Blanks are ignored.

4) Comment material will not affect the M-type translation to Fortran as long as it is to the right of the $ terminator on any line.

Consider example 3, above. To properly use the interpolation subroutine D1DEG1, the user must supply it with three arguments, as follows:

1) The location of the value of the independent variable, x.

2) The location of the integer count for an array of x, y pairs.

3) The location where the result of interpolation, y, is to be placed.

The identifier, RTEST, refers to a specific location in the block of memory reserved for global user constants. The identifier (or symbolic reference form), A5, refers to the location of array number 5. The identifier, G7, refers to the location reserved for the conductance of conductor

number 7. Thus, example 3 illustrates an M-type statement which accomplished the operation: "Call a subroutine which interpolates an array number 5 to find the conductance of conductor number 7, using the value of user constant RTEST as the independent variable". Note that, in using subroutine D1DEG1 to operate on RTEST, A5, and G7, it was not necessary for the user to know, for example, where conductor 7 was located within the block of memory locations reserved for conductances. It was merely necessary for him to indicate, symbolically, that it was specifically conductor number 7 that he wished to evaluate. This *actual* numbering system is unacceptable to the Fortran compiler because the preprocessor reorganizes and packs the user's data, as described in the discussion of the data blocks. However, the preprocessor rectifies this inconsistency by performing a translation on each M-type statement, replacing each *actual* reference with its corresponding, Fortran compatible, *relative* reference.

Example 3A shows the extension of M-type statements to include submodel names in the identifier forms. In this case, the submodel name SUBM1 did not appear in the preceding header record, so the name had to appear as a qualifier with A5 and G7. RTEST is a "global" user constant that is not and cannot be identified exclusively with a particular submodel. If the independent variable was part of a particular submodel, the user constant would have to be numbered, and Example 3 might appear as follows:

```
CALL D1DEG1 (SUBM3.XK12, A5, G7)          $ EX. 3B
```

To generalize on examples 3, 3A & 3B, the following list describes the form of all arguments permitted in M-type statements:

1) Data values which are acceptable to the FORTRAN compiler:
   a. Integer (e.g., 12345678)
   b. Floating point (e.g., 12.4, 4.0E-8)
   c. Character (e.g., 'ABCD')

2) Temperature references of the form Tn or smn.Tn where n = actual node number and smn is a submodel name (e.g., T4,POD2.T4)

3) Capacitance references of the form Cn or smn.Cn (e.g., C6, POD2.C6)

4) Source references of the form Qn or smn.Qn (e.g., Q7, POD2.Q7)

5) Conductance references of the form Gn or smn.Gn where n is an actual conductor number (e.g., G12, POD1.G12)

6) User constant references of the form Kn, XKn, smn.Kn or smn.XKn were n is an actual user constant number (e.g., K11, POD2.XK12)

7) Named global user constant references (e.g., TTEST, KTEST, JOE, MATT)

8) Control constant references (e.g., TIMEND, POD6.DRLXCA)

9) Numeric user array references of the forms An, smn.An, A(n+i), smn.A(n+i), NAn, smn.NAn, NA(n+i) or smn.NA(n+i) where n is an actual user array number and i is the element position in the array (e.g., A6, POD6.A(6+12), NA2)

10) References to temperature at the end of the previous time step, TOLDn and smn.TOLDn (e.g., TOLD112, POD6.TOLD112)

11) References to fluid variables (PL, TL, FR, etc.) as defined in other sections.

If desired the form (n+i) (always within parentheses) may be substituted for n in any of the above forms. Thus, the reference G(120 + 12) would point to the value of the 12th conductor entered following actual conductor number 120 in the conductor data block. References of this form are sometimes useful because the various GEN options store program variables in a strict, known order. The user should realize, however, that using values of i greater than the number of nodes or conductors created by a GEN input is likely to be error prone. Under no circumstances should the user try to use this form to point from one submodel into another. The preprocessor sorts input by submodel so that the input order of groups of nodes or conductors may be meaningless. *Dynamic translation routines like NODTRN, CONTRN, INTLMP, INTPAT, and INTTIE are the preferred methods for such uncertain references.* Use of these routines eliminates concerns regarding storage rules and eliminates errors caused by editing a model.

The versatility of the (n+i) reference form makes the M-type statements take on the versatility of Fortran while preserving the ease of referencing program variables by actual number. This versatility is illustrated in the following examples. Assume, first, that the relative number for actual number 90 is 9. Consider, then, the Fortran expressions generated by the preprocessor for the following expressions that appear in M-type statements:

	M-type Expression	Fortran Expression
(1)	T90	T(9)
(2)	T(90)	T(9)
(3)	NA90+1	NA(9)+1
(4)	NA(90)	NA(9)
(5)	A(90+1)	A(9+1)
(6)	A(90+L)	A(9+L)
(7)	G(90+1)+L	G(9+1)+L
(8)	G(90+1+L)	G(9+1+L)
(9)	GLOBAL.T(90)	T(90)
(10)	GLOBAL.T(90+K90)	T(90+K(9))

The greatest utility of these reference forms lies in the fact that they permit the user to create Fortran statements which reference arrays or sequences of data values without requiring that the user know the relative locations of these values. For example, suppose that a group of ten boundary nodes, numbered 15 through 24, were entered in the NODE DATA block by means of the GEN option. Further suppose that the temperatures for these nodes are to be read from a file tape assigned to logical unit number 3. The following M-type statement will accomplish this action:

```
M     READ(3)(T(15+J),J=0,9)                    $ EX. 6
```

Here, again, it was not necessary for the user to know the relative location of the temperatures, although the knowledge that their relative ordering placed them in sequence, one immediately after the other, proved quite useful.

As another example, the following M-type statement will cause all of the values in array number 90 to be written out on logical unit number 4, regardless of the number of values or the particular *relative* location of the array:

```
M     WRITE (4)(A(90+I),I=1,NA(90))                $ EX. 7
```

Nested references are also useful. Another statement might write out a set of temperatures that vary according to the values of one or more user constants. For instance:

```
M     WRITE(3) (T(15+K12+I), I = K1, K13)          $ EX. 8
```

It can be seen that these reference forms can become extremely complex. The only inflexible rule is that in the sequence: "key variable name-left paren-n" [e.g., "T(n"], n must be a literal integer corresponding to some actual number of an input data value. This integer is always the first following a translatable keyword, and *only this integer will be translated.* Note also that the non-subscripted form (e.g., K12) disallows expressions. The user who inadvertently enters an expression without parentheses, which was allowed in old SINDA-type statements, will be due for a long debug session. A good habit would be to always use parentheses, as required in Fortran, as a means to reduce error potential.

In examples 6, 7 and 8, submodel references were not included to enhance clarity. Any of the key variable names may be preceded by a submodel name qualifier in any M-type statement.

For those cases where access to relative data is required, global access is provided. The input "GLOBAL.key-variable-name (GLOBAL.Tn) turn off the translation of N. This allows translation to be turned off term by term and not just by line as before. Global references are dangerous and should be avoided.

The following statements illustrate the conversion process when the preprocessor generates Fortran expressions from M-type statements. In considering these examples, assume that there is the following *actual/relative* correspondence. Assume for clarity that the same correspondence applies by coincidence to each data block.

ACTUAL (n)	RELATIVE (m)
10	1
20	2
30	3
40	4
50	5

```
      XK20   = (T10+T40)/2.0                    $ EX. 9
42    G10    = SQRT(K20)*XK30                   $ EX. 10
      XK40   = A50                              $ EX. 11
      ATEST  = XK20 + GLOBAL.XK20               $ EX. 11A

C     FORTRAN EQUIVALENTS FOR EXAMPLES 9, 10, AND 11:

      XK(2)  = (T(1)+T(4))/2.0                  EQ. EX. 9
42    G(1)   = SQRT(K(2))*XK(3)                 EQ. EX. 10
      XK(4)  = A(5)                             EQ. EX. 11
      ATEST  = XK(2) + XK(20)                   EQ. EX. 11A
```

Example 9 will cause the average temperature of nodes 10 and 40 to be placed in user constant 20 as a floating point number. The statement in example 10, which has been assigned statement number 42, will compute the conductance of conductor 10 as the square root of the value in user constant 20, times the value in user constant 30. The conductance will be a floating point number, as required, and it is assumed that the user constant 30 contains a floating point number (because the floating point identifier, 'XK', was used in place of the integer identifier 'K'). The Fortran library square root function, SQRT, expects a floating point argument, and it is clear from example 9 that constant 20 does, indeed, contain a floating point number. The array reference, A50, in example 11, is of the integer count form An, and thus references the integer count for array number 50. The statement will result in user constant 40 receiving this integer value. No conversion of arithmetic type will occur because the identifiers, 'A' and 'XK', are both of the same type (i.e., floating point), and the computer's internal logic will not be confused because replacement (i.e., equals sign) is not an arithmetic or relational operation. Quite simply, the content of location A50 (which just happens to be an integer) will be stored in the location of user constant 40. As a result of this statement, the *integer* identifier, K40, may be used in subsequent statements as a reference to the value of the integer-count for array 50.

When using subroutines, the user must take particular care to see that the type of each argument supplied matches the type of each argument expected. That is, if the description of a particular subroutine specifies, for example, that each argument must be a floating point number, the user must be certain that the identifiers that are supplied as arguments refer to locations which contain floating point numbers, and that any data values supplied directly as arguments are, in fact, floating point data values. The following examples illustrate this point. Consider subroutine ADD, which places the floating point sum of the first n-1 floating point arguments into the nth argument. Assume that the following record was entered in a USER DATA block:

```
10=1.2, 11=2.4, 12=0, 13=1.0, 14=1, 15=0, 16=4
```

Example 12, below, shows a correct usage of subroutine ADD. This statement would cause the value 4.6 to be stored in the memory location for user constant number 12. Example 13, however, is incorrect because the second argument does not represent a floating point number.

```
M     CALL ADD   (K10,K13,K11,K12)       $ EX. 12 (correct)
M     CALL ADD   (K10,K14,K11,K15)       $ EX. 13 (incorrect)
```

Subroutine ADDFIX, on the other hand, operates on integer (fixed point) arguments, thus making example 14 correct, with constant K15 receiving the value 5, and example 15 incorrect because K13 is not an integer.

```
M     CALL ADDFIX (K14,K16,K15)          $ EX. 14 (correct)
M     CALL ADDFIX (K13,K16,K15)          $ EX. 15 (incorrect)
```

As explained in the discussion of the USER DATA blocks, an identifier is equated to the memory location where the current value represented by the identifier is stored. When identifiers appear in argument lists, they serve to communicate either (1) the location of a particular value, or (2) the starting location of a sequence of values, depending on what the given subroutine expects. Thus, for example, the identifier T5 may be used as an argument that is expected to be a single floating point value. The value used by the subroutine will be the current value of the temperature of node number 5. In addition, however, this same identifier could be used as an argument that is expected to be the starting location of a sequence of floating point values (i.e., the starting location of an array). The values used by the subroutine will be the sequence of data values located in the node temperature table beginning with the temperature of node number 5.

To illustrate these two uses for the same identifier, consider the following examples. Assume that the following record was included in a NODE DATA block:

```
      GEN 5,10,1,70.0,4.5                $ 10 NODES, 5 THRU 14
```

and the following cards appeared in a Variables 0 block:

```
M     CALL ADD (T5,T6,T7,T8,T9,T10,STEST)      $ EX. 16
M     CALL SUMMARY (6,T5,STEST)                $ EX. 17
```

Example 16 shows a proper usage of the identifier, T5, as a reference to a single value. On the other hand, subroutine SUMARY requires three arguments. (1) the number of values to be added (integer), (2) the starting location of the sequence of values (array) to be added, and (3) the location to receive the result of the addition. Hence, example 17 shows a proper usage of the same identifier, T5, as reference to the starting location of an array. Clearly, the operation in example 16 performs the same action as the operation in example 17.

It should be clear that, in this type of application, although a knowledge of the relative *numbering* is unnecessary, an awareness of the relative *ordering* of the temperatures can be useful. This holds true for the other classes of data, as well.

Supplying an identifier that represents the starting location of a sequence of data values, however, should not be confused with supplying an identifier that represents the location of a user array. In the former case, the number of data values in the sequence must be supplied as a separate argument, but in the latter case, the integer count of the array provides this number and the sequence of data values is assumed to begin at the memory location immediately following the integer count. Thus, while the user is restricted to using identifiers of the form An (never A(n+i)), when a subroutine expects an argument representing an array integer count, the user may supply almost any identifier as an argument to a subroutine which expects an array starting location (illustrated in example 17).

Consider the following illustration: Assume that array 10 contains floating point values. Comparing example 17 with example 18 below, it will be seen that the latter will cause the sum of the values in array 10 to be placed in global user constant UTEST. The array location reference, A10, supplies the number of values in array 10, and the data value reference, A10+1, supplies the starting location of those values. On the other hand, the use of the array location reference in example 19 supplies the number of values in array 10, along with the implicit knowledge (implicit, that is, in the sense that subroutine D1DEG1 expects it to be so) that the array of values begins at the next location in memory immediately following the location of this integer count.

```
M     CALL SUMARY (A10,A(10+1),UTEST)          $ EX. 18
M     CALL D1DEG1 (TTEST,A10,RTEST)            $ EX. 19
```

When using explicit data values (instead of identifiers) as arguments, the user must take great care to insure that the given subroutine does not alter such arguments in any way. Stated another way, any argument which is (or might be) altered by a subroutine must be supplied as an identifier, not a data value. While this rule is seldom violated intentionally, experience has shown that it is often violated through oversight, with generally undesirable consequences. In the usual case, the error is caused by transposing the order of the arguments when calling a particular subroutine. Consider the following statements (see next section for F-type statements):

```
      KTEST=1
      CALL ADDFIX (KTEST,3,2)
      LTEST=2+2
F     WRITE (6,100) LTEST
F 100 FORMAT (I8)
```

The call to subroutine ADDFIX is clearly in error because the result of the addition will be returned to the last argument which is not an identifier. Evidently the user believed that the result would be returned to the first argument. The chaotic effect of this bad call will be obvious when it is understood that the value of LTEST that is printed by the WRITE statement will be 8, not 4! Two plus two equals eight? Yes, because '2' no longer equals 2 after the bad call to ADDFIX; it now equals 4, the sum of KTEST and 3. Hence, the warning is clear: *do not supply data values as arguments that receive the results of a subroutine.*

3.12.3.2 F-Type Statements

An F-type statement is any valid Fortran statement. Such statements may be used in any of the logic blocks, and their basic format is as follows:

```
                                                        7
1     67          Column                                3
|     ||                                                |

Fsn   Fortran-statement                                 comments

F      GO TO 120                                        EX. 20
F140   IF (K(3).LE.75) ITEST = 4                        EX. 21
F130   IF (RTEST.LE.STEST.AND.ITEST                     EX. 22
F     *.GT.KTEST) GO TO 27
```

The 'F' in column 1 must be present in order to inform the preprocessor that the card contains an F-type Fortran Statement. A group of F statements may be blocked using FSTART and FSTOP. A statement number, sn, of one to four digits may be entered in columns 2-5, if required by the user's application. If a complete statement will not fit in columns 7-72, it may be continued in columns 7-72 of succeeding records by placing a "continuation character" (any character except blank or zero) in column 6 of each additional record.

Example 20 shows a correct F-type statement. Example 21 shows a correct statement including a statement number, and example 22 shows how a statement may be extended to more than one record.

Since the dollar sign, $, has a special meaning in Fortran, it may not be used to indicate the end of the statement field on an F-type record. All comments must be restricted to columns 73-80. Although SINDA/FLUINT provides a comments-only card in the form of the REM card, it will also accept the following, Fortran-compatible comment card:

```
C THIS FORTRAN-COMPATIBLE COMMENT CARD IS EXAMPLE 23
```

Note that a different type of user constant reference form, K(3), is included in example 21. This is a Fortran-compatible *relative* reference. It directly addresses the third location in the table of user constants. The constant so referenced will be the third user constant entered in the USER DATA block. *That example assumes the user knows absolutely what to expect in the third location of that array—he has already performed a translation or a DIRECTORIES command, and is willing to take some risk that no changes will take place in the relative storage sequence. Relative* references take the general form of:

```
L(m+i)
```

where:

L ..... the same key letter used for *actual* references (i.e., T,C,Q,G,K,XK, etc)
m ..... *relative* number.
i ..... a *relative* number increment

Since the user constants are referenced by the letter 'K', and 'K' is assumed by the Fortran compiler to represent integer values, a conflict of arithmetic type may result when the location referenced contains a floating point value. For this reason, references to floating point user constants should use the key letters 'XK' in place of 'K'*. (Note that the identifiers 'K' and 'XK' are automatically EQUIVALENCED in each logic block.) The 'XK' form need be used only in arithmetic and relational expressions in Fortran. For example, consider the following statements:

```
F     IF (RTEST.LE.K(3))STEST=K(4)      $ EX. 24 (incorrect)
F     IF (RTEST.LE.XK(3))STEST=XK(4)    $ EX. 25 (correct)
```

If user constants K(3) and K(4) contain floating point numbers, the statement in example 24 will produce erroneous results. The difficulty arises because the Fortran compiler will assume that K(3) and K(4) contain integers, which must be converted to floating point values before being compared with or transferred to the floating point values RTEST and STEST, respectively. Still assuming that K(3) and K(4) contain floating point numbers, then example 25 shows the correct form for referencing them.

Referencing data by its relative input order is clearly much more tedious and prone to error than referencing data by its user-assigned actual number. For this reason, the M-type statement was developed to permit the user to reference data by its actual number within ordinary Fortran-type statements.

There is one application where the relative numbering system may be used. This is the case where the user wishes to reference an entire data array of known size (e.g., the entire array of temperatures, or conductances, etc.). Clearly, it would be preferable to reference the first relative location of the array, rather than to attempt to fix the input order such that node or conductor number such-and-so is always the first element defined in a given block. For example, the following statements will cause the average temperature of all the nodes in a network to be placed in global user constant TTEST:

```
F     CALL SUMARY (NNOD,T(1),STEST)     $ Example 26
F     TTEST = STEST/NNOD                $ Example 27
```

The identifier, NNOD, is among a group of problem size parameters which are initialized by the preprocessor, must never be altered by the user, and may be used in F or M-type statements. NNOD represents the total number of temperatures in the temperature table. T(1) references the first relative location in the temperature table. Hence, it should be clear that the statement in example 26 will cause the sum of all the temperatures in a problem to be placed in global user constant STEST, regardless of the number of nodes or their relative input order in the NODE DATA block. By way of comparison, the same basic action could be implemented with the following M-type statements:

```
M     CALL SUMARY (NNOD,T7,STEST)       $ Example 28
M     TTEST = STEST/NNOD                $ Example 29
```

Note that the statement in example 28 forces a constraint on the input data; namely, that the first node entered in the NODE DATA block must always be node number 7.

Note that example 29 is identical with example 27, above, except for the M. This illustrates that there is no difference in the code resulting from F & M-type statements unless one of the key program variables T, G, Q... TOLD are present. The only drawback to this is that the preprocessor will waste time searching these M-type records for key variables before passing the statement along unchanged to the Fortran compiler.

The above text and examples apply when there is only one submodel defined in the data blocks. The advent of submodels has serious implications on the way the network data is stored in computer memory, and thus the problem of referencing and using the program variables in Fortran statements is complicated to the point where it should probably be avoided except for single-submodel situations and other very special cases. The specific reasons for this are:

1) The T, Q and C values are grouped in memory by submodel and sorted in diffusion — arithmetic — boundary order within these submodel groups. This means that, for example, all the arithmetic node temperatures are not stored contiguously but rather in separated sections of the global T array.* Conversely, fluid data is stored in order tank-junction-plenum and tube-connector (ties are not divided by type), and divided by submodel within these groups.

2) The submodels found in *all* the BUILD records found in the OPERATIONS DATA are represented in the T,Q,C,G,A and K arrays.

This means that, for OPERATIONS DATA blocks with 2 or more build records, parts of the T,Q,C,G,A and K arrays may be inactive part of the time which means that parts of the arrays don't even apply to the current problem. Further, the total number of nodes, NNOD, applies to all the nodes that are ever active, so it may be larger than the number of nodes currently in the active model.

If Fortran statements must be used (or if the user is uncertain of the storage sequence), pointers into the arrays are best obtained using one of the utility subroutines NODTRN, CONTRN, ARYTRN, MODTRN, NUMTRN, MFLTRN, INTLMP, INTTIE, and INTPAT. (See Section 6.) For instance, the location of the temperature, capacitance or Q-source for actual node number 120 of submodel POD2 can be obtained with the following statement:

```
CALL NODTRN ('POD2', 120, ITEST)
```

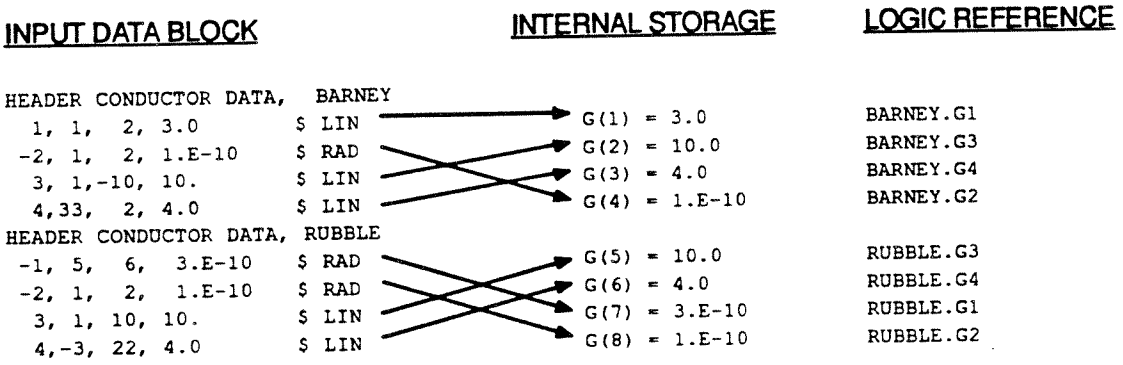
This call will return the relative location of the three attributes of node 120 as an integer number with the variable name ITEST. (Note: the above statement could be either F-type or M-type, since no translation by the preprocessor is required.)

* Analogous FLUINT arrays are stored differently to conserve memory. Lumps are stored in the order tank–junction–plenum with each type subdivided into submodels. Paths are similarly stored as tubes then connectors with each type subdivided by submodel.

3.12.3.3 Examples: Internal Storage vs. Logical Reference

While a user need never know how a parameter is stored internally, experience has demonstrated that advanced users eventually demand such a level of understanding to enable more complex manipulations. This understanding requires a knowledge of translation rules, internal storage sequences, and other such peculiarities. As an aid in putting it all together, this section provides specific multiple–submodel examples of the three types of variables: element–dependent, submodel–dependent, and array–dependent.

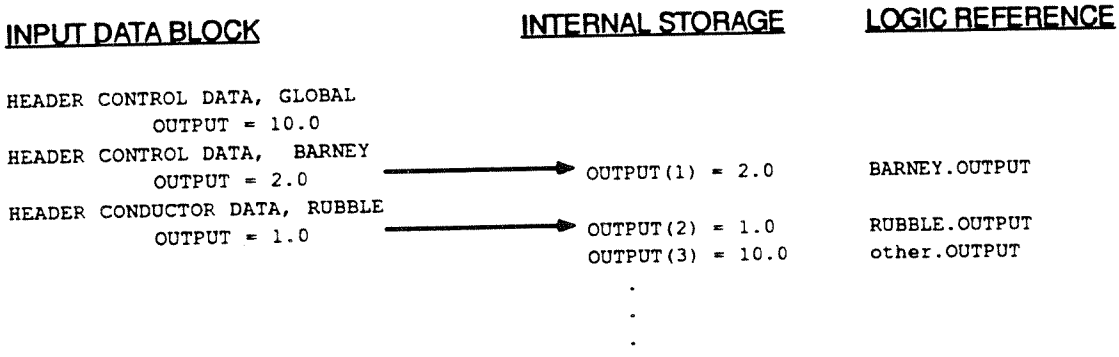
To illustrate element–dependent parameter storage and access, conductors will be used. Recall that conductor data are stored by input submodel order, subdivided by linear and radiation types. Node data are stored analogously, subdivided by diffusion, arithmetic, and boundary. Submodels are stored by input order (if ever built) according to the sequence of NODE DATA and FLOW DATA headers within the input file. The following example illustrates how input order, storage order, and logic references are related for conductors in two submodels:



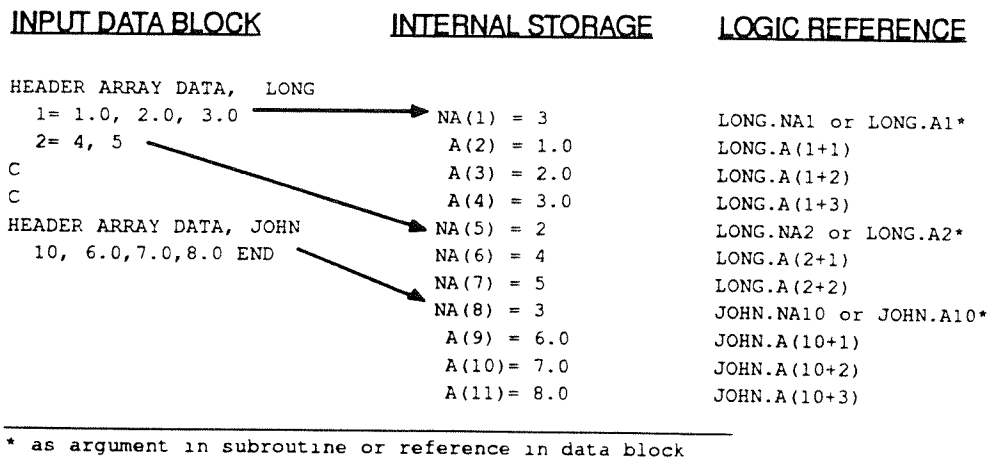
In the above example, HEADER NODE DATA, BARNEY is assumed to precede HEADER NODE DATA, RUBBLE.

Fluid submodel element–dependent data follow similar rules to the above example, except they are grouped first by type (tank, junction, and plenum, or tube and connector). Within the one category (e.g., tanks), they are subdivided by submodel.

Submodel–specific control constants, whether thermal or fluid, are stored internally as one dimensional arrays whose subscript refers to the submodel input order. The following example illustrates how input order, storage order, and logic references are related for control constants in the same two thermal submodels from the last example, assuming they are the first models input:



Array data (excluding TARRAY and CARRAY data) is all stored in one array, again according to submodel order. Unlike other data, a single cell is inserted at the beginning of each array which is equivalenced to its integer length:



* as argument in subroutine or reference in data block

3.12.3.4 The Debug Option

One of the most troublesome errors a SINDA/FLUINT user can make is to misspell a variable name within the logic blocks. Because this results in a variable being defined incorrectly, the error may never be really visible or may turn up much later in the run, usually in an unpredictable and catastrophic fashion.

SINDA/FLUINT offers a debug feature which can be quite helpful. The debug option takes advantage of the fact that a spelling error will invariably create a so-called local variable. In SINDA/FLUINT, the preprocessor defines a local variable as one whose name does not match any program control constant, named user constant, numbered user constant, array element or reference, or any of the key program variable names (e.g., T,C,Q,G, etc.). *When debug is on, all variables within M-type statements are checked* against the list of variable names and key word/actual number combinations that are defined in the data blocks. When a local variable is identified, an error message will be printed just after the record containing the local variable. For instance, the statement:

```
M     J = K50
```

will generate a preprocessor error message if J is not declared as a global user constant. If user constant 50 did not appear in a USER DATA block, an additional error would result.

It can be seen that the difficulty with this feature is that the user can frequently generate valid local variables that are not misspelled, or the user may simply inactivate debug in order to eliminate annoying messages. The solution to this requires the user to get in the habit of declaring all of the variable names planned for use in M-type statements in the global USER DATA block. Users familiar with programming languages other than Fortran will recognize this as a firm requirement. Fortran, however, does not enforce this discipline. If these variables are all in the USER DATA, the debug option will only point to misspelled local variables and keyword variables that have not been declared properly or were misspelled, either in the statement that triggered the error or in the data block declaration.

If the user takes no action, the debug feature will be on by default. The DEBON and DEBOFF macroinstructions provide local control of debug. A DEBOFF appearing in a logic block will turn off debug until the next logic header record is encountered. DEBOFF reverses DEBON. NODEBUG in the OPTIONS DATA block turns off debug for all logic blocks, although it can be turned back on locally with a DEBON instruction.

It should be emphasized that the debug option does not apply to F-type statements. This is because F-type statements are not examined character by character, but rather they are passed on to the compiler unchanged. Since in the majority of cases, an M-type statement can be written that is equivalent to an F-type, it is recommended that M-type be used as much as possible in large, complex models that are likely to benefit from this debug feature.

3.13 FLUID PROPERTY (FPROP) DATA

3.13.1 General Information

Fluid submodels require the complete description of the working fluid's properties. This description may be provided in one of two ways:

1) by naming one of 20 *standard library fluids* using the appropriate ASHRAE number (see Appendix C) on the HEADER FLOW DATA record, or

2) by creating an FPROP DATA block defining a new *user-defined fluid*, and then naming that fluid on any HEADER FLOW DATA record.

A user-defined fluid may be used as the working fluid in any, all, or none of the fluid submodels. In addition, the user may refer to that fluid in any relevant property routine (see Section 6.9.2) in any logic block. The model does not have to contain any fluid submodels to be able to make such property calculations; the FPROP DATA blocks can be used merely as a convenience in thermal submodels.

FPROP DATA blocks were designed such that they can be easily reused in any input file without worrying about unit system conversions. The work involved in defining and debugging a fluid description need only be performed once. The user is encouraged to maintain his own library of alternate fluid descriptions in separate files which can be added to any SINDA/FLUINT input file using the INCLUDE command. A special output routine PRPMAP is available to assist in preparing and debugging a fluid description (see Section 6), and an external program named RAPPR (see Section 7.3) exists which can be used to create custom and fast but simplified FPROP DATA blocks from standard library fluids.

Four basic types of user-defined fluids are allowed:

1) A perfect gas (that cannot condense). Both the specific heat and transport properties may vary with temperature—the gas does not have to be calorically perfect. The fluid identifiers for these fluids can range from 8000 to 8999.

2) A simple liquid (that cannot boil). The liquid is assumed to be incompressible, but density as well as transport properties can vary with temperature. The fluid identifiers for these fluids can range from 9000 to 9999.

3) A simplified description of a two-phase fluid. This option is somewhat like a combination of the above two fluids, with saturation dome and two-phase properties also required. This option is intended to allow a limited description of a two-phase fluid to be quickly added using readily available property data. The fluid identifiers for these fluids can range from 7000 to 7999.

4) An advanced (complete) description of a two-phase fluid or real gas. The user may replace some of all of the standard library property routines with alternates; in other words, the user may input a functional description of a fluid over a very wide range of properties, *including supercritical if desired.* This option is really more of a logic block than a data block, and is used by RAPPR to generate simplified, localized versions of the 20 standard library fluids. Also, fast tabular descriptions of water, ammonia, hydrogen, nitrogen, oxygen, ethane, argon, and others have been created using this option. The fluid identifiers for these fluids can range from 6000 to 6999.

The ability to define new fluids significantly expands the usefulness of the program by allowing simulation of more fluids (such as nitrogen, water/glycol, toluene, and air) over a wider range of temperatures and pressures. Furthermore, the program runs an order of magnitude faster using a simplified description of a standard library fluid (such as liquid water). Refer also to Section 4.7.15, and to Section 7.3 (RAPPR).

Each FPROP DATA block contains the complete description for one working fluid. Up to 10 FPROP DATA blocks may appear anywhere in the input stream after OPTIONS DATA.

The following four subsections apply only to 7000, 8000, and 9000 series fluids. The exception is 6000 series (advanced two-phase), which are really logic blocks that require completely different input rules.

3.13.2 FPROP DATA Blocks

FPROP DATA blocks are composed of two types of subblocks. The first kind is defined by the header record itself, which may extend across several lines of input. It is called the *header subblock*, and every FPROP block has one and only one such subblock. The second kind, the *array subblock*, is identified by an "A" in column 1. The number of array subblocks within an FPROP DATA block is variable and may be zero. Data entry in each subblock continues until a new array subblock or header record is encountered. The rules for data entry within each subblock are described in the next subsections.

There are two ways of describing a property in an FPROP DATA block. The first is *reference style*, where the user provides a reference temperature, the value of a property at that reference temperature, and a temperature coefficient (which may be zero). The general formula for calculating such a property as a function of temperature is then:

$$X(T) = X + (T - TREF)*XTC$$

where

X(T) ..... the calculated value of property X at temperature T
X ..... the input reference value of the property at temperature TREF
TREF ..... the input reference temperature
XTC ..... the temperature coefficient for X (defaulted to zero meaning constant properties); positive if the property increases with increasing temperature.

There is only one reference temperature input per phase; all properties described in this style must be consistent with this requirement. All reference style descriptions are contained within the HEADER FPROP DATA subblock.

The second method for describing properties is *array style* using a bivariate array of the property versus temperature. Each array style property is input using a separate array subblock, as described below. If both reference style and array style inputs are used for the same variable, a caution is printed and the latest input, always an array subblock, is used.

Fortran style calculations (*,/,+,-) are allowed although expressions are evaluated left to right with no precedence given to * or /, and no functions or parentheses are allowed. User constants may not be referenced within these data blocks.

3.13.2.1 Header Subblocks

The header subblock in an FPROP DATA block can be used to supply the fluid name (which implies the fluid type), the local unit systems, operating range limits, and any reference style data. The format is:

```
HEADER FPROP DATA, Nnnn [,uid [,abszro] ]
     [,keyword=value][,keyword=value]
```

where:

`Nnnn` ..... the **four** digit number assigned by the user to the fluid about to be described. The first digit (N) must be either an 8 (meaning perfect gas), a 9 (meaning incompressible liquid), a 7 (meaning simplified two-phase), or a 6 (meaning advanced two-phase). (No default.)

`uid` ..... local unit system: SI or ENG (see CONTROL DATA, GLOBAL). Defines the *local* unit system of *this and only this* HEADER block. (Default: value input in CONTROL DATA)

`abszro` ... absolute zero in the local unit system (see CONTROL DATA, GLOBAL). Defines the *local* unit system of *this and only this* HEADER block. (Default: value input in CONTROL DATA if uid not given, otherwise -459.67 or -273.15 according to the uid value input)

`keyword` .. a keyword appropriate to the fluid type being input, such as 'X' and 'XTC' where X is a property name

`value` .... a value appropriate to the previous keyword

Together, the values of uid and abszro define the unit system used within the FPROP DATA block. All data within the block must then be consistent with this chosen system. For example, if uid=ENG and abszro=-460.0, then all temperatures are expected to be in °F, and thermal conductivity is expected in BTU/hr-ft^2-°F. Note that both abszro and uid are optional, but uid *must* be input if abszro is to be input. *It is strongly recommended that the user input* both *uid and abszro.* This not only avoids confusion, it makes FPROP DATA blocks transportable to any other model without unit system collisions. Note that if pressures are required, they are always assumed to be in absolute units.

The remaining inputs in the subblock are dependent on the fluid type.

3.13.2.2 Array Subblocks

Array subblocks are used to input bivariate arrays of a property versus temperature. The format is:

```
AT, X, t1,v1, t2,v2
     ... ,ti,vi, ... tn,vn
```

where:

X ..... property name
t,v ..... temperature/value data pairs

The temperature values must rise monotonically. AT blocks are interpolated within the array limits and extrapolated beyond them. A global maximum of 2000 data points may be used in any one model (e.g., 1000 data pairs).

Remember, all reference style inputs must occur *within* the subblock formed by the HEADER FPROP DATA command, and all AT subblocks occur *after* the HEADER subblock. The appearance of an AT subblock will override any reference style input for the same property.

3.13.3 Perfect Gas (8000 Series)

A perfect gas is completely described by four quantities: a gas constant, a specific heat at constant pressure, a thermal conductivity, and a dynamic viscosity. The gas constant is given by $\mathcal{R}$/MW, where $\mathcal{R}$ is the universal gas constant (8314.34 J/kgmol-K or 1545.33 ft-lb/lbmol-R) and MW is the molecular weight. The remaining properties may vary with temperature and should be provided in standard units. Refer to Section 4.7.15 and Appendix D for more information, and Section 7.3 for more examples.

All input properties must be positive. The default temperature range for fluid submodels using a perfect gas description is 10^{-3} to 10^{10}, *or that range over which all input properties are positive.* The program will print an advisory message describing any such limitations at the start of processor execution. (See Appendix C.)

The formats within the header subblock are:

```
RGAS=R [,TREF=R]
[,TMIN=R] [,TMAX=R]
{,K=R}   [,KTC=R]
{,CP=R}  [,CPTC=R]
{,V=R}   [,VTC=R]
```

where:

RGAS ..... the gas constant. (No default.)

TREF ..... the reference temperature for all reference style properties. This is not needed if the all the properties are constants or if they are all input as arrays. (Default: 0.0 in *local* temperature units. It is strongly recommended that the TREF value be stated explicitly.)

TMIN ..... the minimum allowable temperature. (Default: 10^{-3} in *absolute* temperature units, or the minimum implied by other inputs.)

TMAX ..... the maximum allowable temperature. (Default: 10^{10} in *absolute* temperature units, or the minimum implied by other inputs.)

K ..... the value of thermal conductivity at TREF. Must be supplied if AT,K subblock is not. (No default.)

KTC ..... thermal conductivity temperature coefficient. (Default: 0.0, constant thermal conductivity.)

CP ..... the value of specific heat at TREF. Must be supplied if AT,CP subblock is not. (No default.)

CPTC ..... specific heat temperature coefficient. (Default: 0.0, constant specific heat.)

V ..... the value of dynamic viscosity at TREF. Must be supplied if AT,V subblock is not. (No default.)

VTC ..... dynamic viscosity temperature coefficient. (Default: 0.0, constant dynamic viscosity.)

There are three optional AT subblocks, one for each temperature-varying property:

```
{AT,K, ...}
{AT,CP, ...}
{AT,V, ...}
```

Example 8000 Series FPROP DATA Blocks:

```
HEADER FPROP DATA,8001,ENG,-460.0            $ GAS 8001
       RGAS          = 1525.0
       V             = 1.0E-4,   VTC = -1.0E-6
       K             = 22.0,               TREF = 100.0
AT,CP,22.0,0.1,      33.0,0.115,100.0,0.15
C
HEADER FPROP DATA,8888,SI                    $ abszro defaults
       CP=2.0, V=1.E-3, K=0.1, RGAS=55.0
C
HEADER FPROP DATA,8717
       TMIN          = 50.0,               TMAX       = 1000.0
       RGAS          = 8313.34/18.0,       TREF       = 100.0
       V             = 1.0 + 2.0,          VTC        = -0.004
AT, K, 100.0,        12.0,
         200.0,      14.0,
         300.0,      15.0
AT, CP, 0.0,0.2,     200.0,0.21,1000.0,0.19
```

3.13.4 Incompressible Liquid (9000 Series)

An incompressible liquid is completely described by four quantities: a specific heat at constant pressure, a density, a thermal conductivity, and a dynamic viscosity. All of these properties may vary with temperature and should be provided in standard units. Refer to Section 4.7.15 and Appendix D for more information, and Section 7.3 (RAPPR) for more examples.

All input properties must be positive. The default temperature range for fluid submodels using an incompressible liquid description is 10^{-3} to 10^{10}, *or that range over which all input properties are positive.* The program will print an advisory message describing any such limitations at the start of processor execution. (See Appendix C.)

The formats within the header subblock are:

```
[,TREF=R]
[,TMIN=R] [,TMAX=R]
{,K=R}    [,KTC=R]
{,CP=R}   [,CPTC=R]
{,V=R}    [,VTC=R]
{,D=R}    [,DTC=R]
```

where:

TREF ..... the reference temperature for all reference style properties. This is not needed if the all the properties are constants or if they are all input as arrays. (Default: 0.0)

TMIN ..... the minimum allowable temperature. (Default: 10^{-3} 0.0 in *absolute* temperature units, or the minimum implied by other inputs.)

TMAX ..... the maximum allowable temperature. (Default: 10^{10} in *absolute* temperature units, or the minimum implied by other inputs.)

K ..... the value of thermal conductivity at TREF. Must be supplied if AT,K subblock is not. (No default.)

KTC ..... thermal conductivity temperature coefficient. (Default: 0.0; constant thermal conductivity.)

CP ..... the value of specific heat at TREF. Must be supplied if AT,CP subblock is not. (No default.)

CPTC ..... specific heat temperature coefficient. (Default: 0.0; constant specific heat.)

V ..... the value of dynamic viscosity at TREF. Must be supplied if AT,V subblock is not. (No default.)

VTC ..... dynamic viscosity temperature coefficient. (Default: 0.0; constant viscosity.)

D ..... the value of density at TREF. Must be supplied if AT,D subblock is not. (No default.)

DTC ..... density temperature coefficient. (Default: 0.0, constant density—no thermal expansions/contractions will be considered.)

There are four optional AT subblocks, one for each temperature-varying property:

```
{AT,K, ...}
{AT,CP, ...}
{AT,V, ...}
{AT,D, ...}
```

Example 9000 Series FPROP DATA Blocks:

```
HEADER FPROP DATA,9711,SI,0.0     $ LIQUID 9711
       TMIN           = 100.0,              TMAX       = 1000.0
       TREF           = 400.0,                         $ REF. TEMP
       CP             = 1004.0*0.001                   $ SP. HEAT
       D              = 60.0, DTC=-0.001               $ DENSITY
AT,K,0.0,100.0                                         $ COND. ARRAY
       300.0, 50.0
       500.0,30.0, 700.0,40.0
AT,V,200.0,13.0E-6                           $ VISC. ARRAY
C
C THE NEXT BLOCK DEFAULTS uid AND abszro:
C
HEADER FPROP DATA,9333,K=1.0,CP=2.0,D=3.0,V=4.0
```

3.13.5 Simplified Two-Phase Fluid (7000 Series)

3.13.5.1 Introduction

This input option allows the user to describe an alternate two-phase working fluid using parameters readily found in common reference materials. Fluids input using this method must be numbered between 7000 and 7999.

Although the range limits are somewhat narrow because of the simplifying assumptions, the properties are reasonably accurate for preliminary design work. This section documents the methods used for generating consistent thermodynamic properties given a minimum of simple inputs.

The liquid density, gas specific heat, and all transport properties may be input as constants or as arrays with temperature (i.e., the same method used for 8000 and 9000 fluids). The generation of the P-v-T and P_{sat}-T_{sat} relations require more work because such data is not typically available in reference sources and yet must be derived on the basis of data that *is* available. Also, the analytic form of these relations must be known by the program because of speed considerations.

For the P-v-T surface, the van der Waal's equation is used:

$$P = \frac{RT}{v-b} - \frac{a}{v^2}$$

Although some handbooks list constants a and b, they can be calculated given the critical point for the fluid. This is the best general P-v-T relation that can be calculated on the basis of simple user-provided properties. However, the approximation is not good near or above the critical pressure, and might be inadequate for liquid metals and highly polar molecules.

For the P_{sat}-T_{sat} relation, the following general equation is used:

$$\ln P = A + B/T + C\ln T + DT + ET^2$$

This relationship is adequate for most fluids over a limited range. The five constants in the equation are calculated by the program on the basis of the critical point, and two values each of saturation pressure and heat of vaporization at given saturation temperatures.

FLUINT uses these data and relationships to derive all other required data, including enthalpies and entropies. Thus, using only a few values available in handbooks, the user may adequately describe a two-phase fluid.

The approximations are all very good for nonpolar substances at low gas pressures ($P << P_{crit}$). This makes them especially appropriate for cryogens such as nitrogen, hydrogen, and oxygen. For analyses near or above the critical point, a more detailed description is required—refer to the next subsection on 6000 series fluids. Because the above methods are more accurate for a gas than a perfect gas assumption at all points, the user may wish to use this option in place of

the 8NNN option for single-phase gas analyses. However, the user should avoid this option for single-phase liquid analyses and should use instead the 9NNN option. For liquids, the methods used in the 7NNN series fluids are both more expensive and less accurate than those used in the 9NNN series. (Also, the 7NNN series should never be viewed as an alternative to the prestored two-phase library of 20 fluids—refer to RAPPR in Section 7.)

In fact, *the property with the least accuracy is the specific heat for liquids*, which can differ from the true value by as much as 50% for certain inputs. Thus, a good rule of thumb for checking the accuracy of the description for a given fluid is to check the values given by VCPF with tabulated values. For such a simplified description, differences on the order of 10% can be considered an excellent match. The user should not be overly concerned with such large errors in most two-phase analyses because sensible heating is small compared to the heat of vaporization. For example, Sample Problems C and E in Appendix F produce nearly identical results whether run with library or very simple 7NNN descriptions of water and ammonia (both highly polar molecules). This is true even though the specific heats appear to be in error.

In order to enable some options (including choked flow detection, compressed liquid density, and liquid and two–phase speeds of sound) another constant must be input. This constant, ω_{SRK}, is the acentric factor for each fluid, modified as needed to make the Soave–Redlich–Kwong equation of state match the saturation pressure.

A definitive source for the values of ω_{SRK} for over 400 compounds is *The Properties of Gases and Liquids, Fourth Edition*, by Robert C. Reid et al. The appendix of that same reference also lists values of the true acentric factor, ω, for over 800 fluids. The true acentric factor is defined as $\omega = -\log_{10}(P_{sat}/P_{crit}) - 1.0$, where the saturation pressure P_{sat} is evaluated at a reduced temperature (T/T_{crit}) of 0.7. A subset list of ω_{SRK} for some common fluids, taken from the above reference, is given in Table 3–12. While water, ammonia, and propane are available as library fluids, they are listed for comparison purposes.*

If the desired fluid is not listed in Table 3–12 and the above reference cannot be found, then the true acentric factor for the desired fluid should be input as a good first guess. Thus, *if no value of WSRK is given, FLUINT will calculate the acentric factor* based on the saturation curve

Table 3–12 Modified Acentric Values (WSRK) for Selected Fluids

Fluid	ω_{SRK}	Fluid	ω_{SRK}	Fluid	ω_{SRK}
Water	0.3852	Helium	–0.4766	Acetone	0.3149
Hydrogen†	–0.2324	Neon	–0.0362	Methanol	0.5536
Oxygen	0.0298	Argon	–0.0092	Ethanol	0.6378
Nitrogen†	0.0358	Krypton	–0.0050	Isopropanol	0.6637
Air	–0.0031	Xenon	–0.0023	Methane	0.0074
Carbon Monoxide	0.0295	Flourine	0.0493	Ethane	0.0983
Carbon Dioxide	0.2373	Chlorine	0.0822	Propane	0.1532
Sulfur Dioxide	0.2645	Kerosene	0.4783	n–Butane	0.2008
Ammonia	0.2620	Fuel Oil	0.6009	Isobutane	0.1825
Hydrazine	0.3410	Dowtherm A	0.4084	n–Octane	0.3998
Nitrogen Dioxide	0.8634	Carbon Tet (CCl_4)	0.1875	Ethylene	0.0882
Nitrogen Tetroxide	0.8573	Formaldehyde	0.2656	Propylene	0.1455

† Heed the warning below regarding accuracy of speeds of sound for these fluids.

* In fact, the library descriptions use ω_{SRK} internally. However, as a caution, note that the internal library descriptions of water and ammonia are slightly more complicated to give better accuracy for those aberrant fluids.

that is calculated as a best fit to the input data if a reduced temperature of 0.7 is within the valid range of the fluid (which is nearly always the case). Otherwise, an error message might result from attempts to call or use routines that need this data: VDLC, VSOSF, VSOSFV, and the choke check routines CHKCHL and CHKCHP.

Warning: This approach is simple but approximate, and was chosen because it could encompass the widest range of fluids with the least input—perhaps even none. While the compressed liquid densities are reasonably accurate for all fluids, the corresponding liquid speeds of sound are much less accurate for some fluids. The problem is most acute for cryogens near the saturation line, in which case the liquid speed of sound is overpredicted (because the compressibility is underpredicted), some times by as much as a factor of two or three. The accuracy is good at low temperature and high pressures, but poor at higher temperatures near the saturation line. This problem has been found to be most acute for nitrogen and parahydrogen, but oxygen is curiously immune. Fortunately, the liquid speed of sound contributes little to the two–phase speed of sound except at extremely low qualities. In fact, *the speeds of sound in liquid are almost completely irrelevant for choking calculations even if the local fluid is subcooled liquid*, because in order to choke the flow must first be expanded into a throat, which will almost always result in a two–phase state within the throat except at inordinately high pressures. Still, the user should apply caution in using the VDLC routine to calculate liquid state compliances or local sound speeds. For cryogens, faster and more accurate tabular descriptions are available as 6000 series FPROP blocks.

3.13.5.2 Input Formats

The input rules for this option are very similar to the 8NNN and 9NNN FPROP options, with data required for each phase as well as for saturation properties.

Most values may be input as constants, linearly varying constants, or arrays vs. temperature in the same style as that used in 8NNN and 9NNN FPROP blocks. The only new style of inputs is the *mandatory* AT,DOME section, where two data points are entered to determine the saturation dome. Also, two reference temperatures (one for each phase) may be input instead of one, and range limits may be explicitly defined. Unlike temperatures, *all pressures are assumed to be in absolute units.* All units must be in the unit set named on the HEADER card (see Appendix D).

The formats within the header subblock are:

```
HEADER FPROP DATA, 7nnn, [,uid [,abszro] ]
     RGAS=R, TCRIT=R, PCRIT=R [,TREFG=R] [,TREFL=R]
     {,KG=R} [,KGTC=R]          {,KL=R} [,KLTC=R]
     {,VG=R} [,VGTC=R]          {,VL=R} [,VLTC=R]
     {,CPG=R} [,CPGTC=R]        {,DL=R} [,DLTC=R]
     {,ST=R} [,STTC=R]
     [,AVDW=R] [,BVDW=R]
     [,TMIN=R] [,TGMAX=R] [,PGMAX=R]
     [,WSRK=R]
```

where:

RGAS ..... the gas constant. (No default.)

TCRIT .... critical temperature. (No default.)

PCRIT .... critical pressure. (No default.)

TREFG .... the reference temperature for all GAS properties that will be input as reference values and coefficients. This is not needed if the all the gas properties are constants (XTC=0.0) or if they are all input as arrays. (Default: 0.0 in *local* temperature units. It is strongly recommended that the TREFG value be stated explicitly.)

TREFL .... the reference temperature for all LIQUID properties that will be input as reference values and coefficients. This is not needed if the all the liquid properties are constants (XTC=0.0) or if they are all input as arrays. (Default: 0.0 in *local* temperature units. It is strongly recommended that the TREFL value be stated explicitly.)

KG ..... the value of gas thermal conductivity at TREFG. Must be supplied if AT,KG subblock is not. (No default.)

KGTC ..... gas thermal conductivity temperature coefficient. (Default: 0.0, meaning gas thermal conductivity is constant.)

KL ..... the value of liquid thermal conductivity at TREFL. Must be supplied if AT,KL subblock is not. (No default.)

KLTC ..... liquid thermal conductivity temperature coefficient. (Default: 0.0, meaning liquid thermal conductivity is constant.)

VG ..... the value of gas dynamic viscosity at TREFG. Must be supplied if AT,VG subblock is not. (No default.)

VGTC ..... gas dynamic viscosity temperature coefficient. (Default: 0.0, meaning gas dynamic viscosity is constant.)

VL ..... the value of liquid dynamic viscosity at TREFL. Must be supplied if AT,VL subblock is not. (No default.)

VLTC ..... liquid dynamic viscosity temperature coefficient. (Default: 0.0, meaning liquid dynamic viscosity is constant.)

CPG ..... the value of the gas specific heat at TREFG *(low pressure value)*. Must be supplied if AT,CPG subblock is not. (No default.)

CPGTC .... gas specific heat temperature coefficient. (Default: 0.0, meaning gas specific heat is constant.)

DL ..... the value of saturated liquid density at TREFL. Must be supplied if AT,DL subblock is not. (No default.)

DLTC ..... liquid density temperature coefficient. (Default: 0.0, meaning liquid density is constant and no thermal expansions/contractions will be taken into account.)

ST ..... liquid to vapor surface tension at TREFL. Must be supplied if AT,ST subblock is not. (No default.)

STTC ..... surface tension temperature coefficient. (Default: 0.0, meaning surface tension is constant.)

AVDW ..... Van der Waals coefficient A. This parameter has units of pressure times density squared. In English units, AVDW has units of psf-ft^6/lb$_m$^{2}. (De-

fault: calculated by program.)

BVDW ..... Van der Waals coefficient B. This parameter has units of specific volume. (Default: calculated by program.)

TMIN ..... lowest allowed temperature. May be overridden by program. (Default: calculated by program or 10^{-3} minimum. Lowest allowed pressure is the saturation pressure at this value.)

TGMAX .... highest allowed gas temperature. May be overridden by program. (Default: calculated by program, 1000*TCRIT or 10^{10} maximum.)

PGMAX .... highest allowed gas pressure. May be overridden by program. (Default: PCRIT or 10^{10} maximum. Highest allowed liquid temperature is the saturation temperature at this value.)

WSRK ..... Modified acentric factor. (Default: calculated using the saturation curve generated on the basis AT, DOME inputs)

There is one mandatory and seven optional AT subblocks, one for each temperature-varying property described above:

```
AT,DOME,t1,p1,hfg1,t2,p2,hfg2
{AT,KG, ...}
{AT,KL, ...}
{AT,VG, ...}
{AT,VL, ...}
{AT,CPG, ...}
{AT,DL, ...}
{AT,ST, ...}
```

where:

t1/2 ..... temperature values at which dome properties will be given

p1/2 ..... saturation pressure at temperatures t1, t2

hfg1/2 ..... heat of vaporization at temperatures t1, t2

3.13.5.3 Guidance and Restrictions

Liquid Properties—Note that the C_p of the liquid is never input—it is calculated on the basis of other inputs. Note also that the surface tension is treated as a liquid property, and that the liquid density is for saturated liquid. Liquids are assumed incompressible; the properties are independent of pressure. Also, be warned that liquid enthalpies are frequently negative. This is irrelevant since only enthalpy *differences* are important.

Gas Properties—It is important that the C_p of the gas is the value at zero or very low pressure. This value is sometimes denoted $C_p°$ in tables. As for the remaining gas values (KG and VG), a decision should be made between saturated values and superheated values at constant pressure since these properties are assumed to be independent of pressure. If the analysis includes a variety of pressures with low superheating, the saturation values should be input and an upper temperature limit for the gas should be input. If the analysis takes place at primarily one pressure, the values at that pressure should be input and an upper limit on gas pressure should be input. (Refer to the last example in the next subsection.)

Dome Properties—The saturation dome is characterized by the critical point and two pairs of pressure and heat-of-vaporization values. These last values are input in the mandatory AT,DOME subblock. The choice of which temperature values to use deserves some thought. In general, the temperatures should bracket the range of expected saturation values, *but values at lower temperatures will yield better results than those at higher temperatures.* A good rule of thumb might be to take one value at the lowest available temperature (perhaps just above the freezing point), and the second value at about the temperature anticipated in the analysis. Values near the critical point should be avoided.

Implicit Range Limits—This option is restricted to low pressures compared to the critical point for gases, whereas the pressures in the liquid phase may be arbitrarily high. Conversely, the gas temperature may be arbitrarily high, whereas the liquid temperature is limited by the saturation value at the highest allowed gas pressure. As a rule of thumb, the gas pressure should never exceed one third of the critical value. This limits the liquid temperature to about $0.8T_{CR}$.

While the user may explicitly state the range limits, FLUINT performs additional data checks and may further decrease the valid range to maintain consistency and avoid illegal property values. Note that the program can only avoid *illegal* properties; it is the user's responsibility to avoid *inaccurate* properties by explicitly reducing the range limits. The final set of range limits is enforced by the program, and may be accessed by the user by calling VTMIN, VTGMAX, VTLMAX, and VPGMAX. Note that slight tolerancing is used to avoid the limits, which are assumed to define the valid range exclusively (i.e., the limits themselves represent invalid states).

The top half of Figure 3–18 illustrates the range limits for 7NNN fluids.

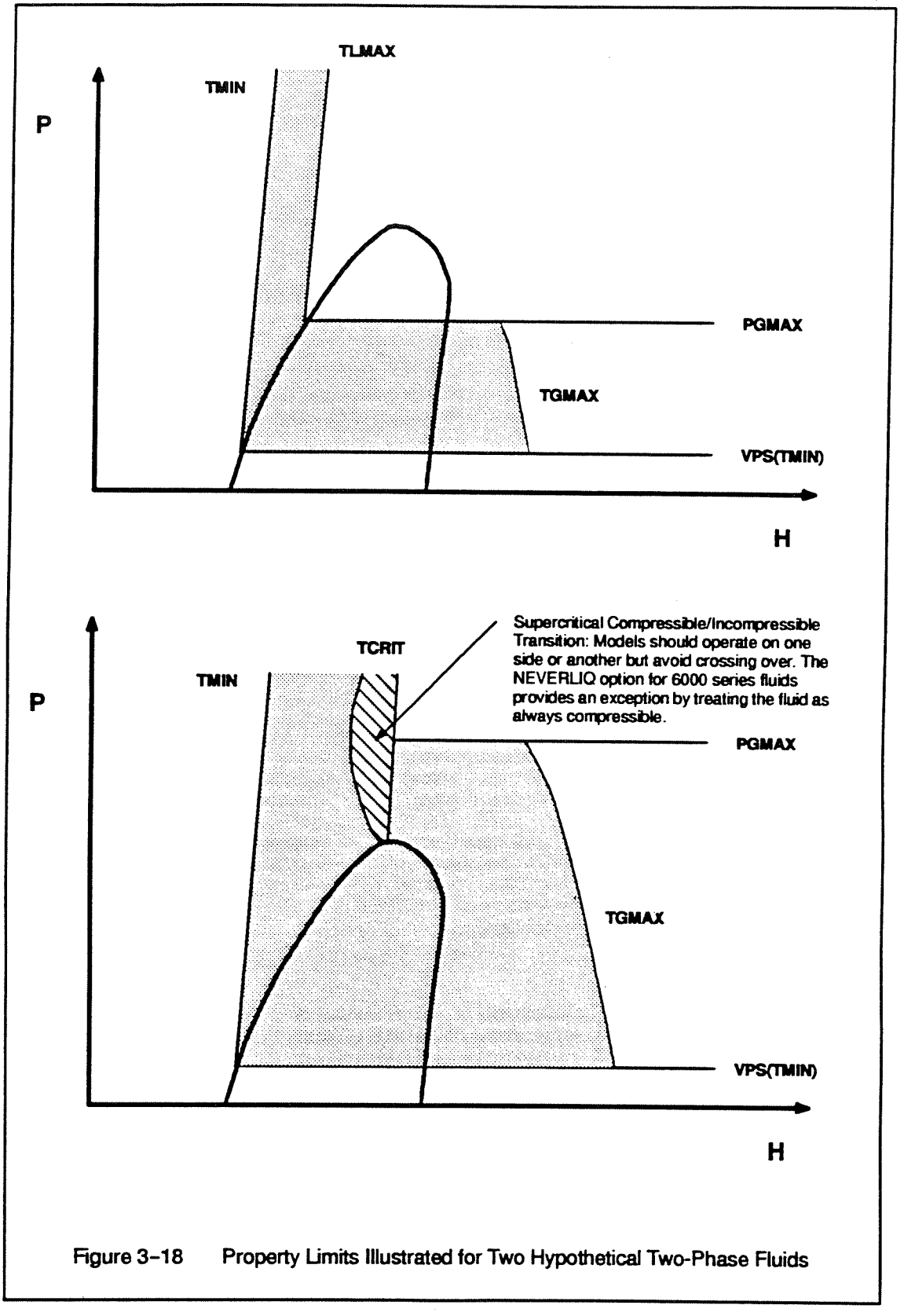


Figure 3-18 Property Limits Illustrated for Two Hypothetical Two-Phase Fluids

3.13.5.4 Example 7000 Series FPROP DATA Blocks

SAMPLE 7nnn FPROP DATA BLOCK:

```
HEADER FPROP DATA,7001,SI,0.0
        TREFG           = 400.0,    TREFL       = 200.0
        DL              = 100.0,    DLTC=-0.001
        RGAS            = 1525.0
        VG              = 1.0E-4,   VTC = -1.0E-6
        KG              = 22.0
        ST              = 0.18,     STTC = -.0035
        TCRIT           = 600.0,    PCRIT = 1.0E8
        TGMAX           = 800.0,    TMIN = 100.0,           PGMAX = 1.0E8
AT,KL,  0.0,100.0
        300.0, 50.0
        500.0,30.0
AT,VL,  200.0,13.0E-6
AT,CP,  22.0,0.1,       33.0,0.115,             100.0,0.15
AT,DOME,                200.0,1.0E5,2.0E4,      300.0,1.0E7,1.0E4
```

SIMPLIFIED FPROP DATA BLOCK FOR NITROGEN:

```
HEADER FPROP DATA,7728,SI,0.0
C
C SIMPLIFIED N2 TWO-PHASE (FROM 63 TO 104K, NEAR 80 K)
C VAPOR PROPERTIES ARE FOR SAT VAPOR
C (EXCEPT CP WHICH MUST BE FOR LOW PRESSURE GAS)
C
        RGAS            = 8314.34/28.01
        TCRIT           = 126.2
        PCRIT           = 3.4E6
        TREFG           = 80.0
                        CPG         = 1.039E3,  CPGTC       = 0.0
                        KG          = 7.7E-3,   KGTC        = 0.125E-3
                        VG          = 5.6E-6,   VGTC        = 0.076E-6
        TREFL           = 80.0
                        VL          = 148.0E-6, VLTC        = -5.35E-6
                        KL          = 132.2E-3, KLTC        = -1.84E-3
                        DL          = 796.24,   DLTC        = -4.739
                        ST          = 8.27E-3
                        WSRK        = 0.0358
        TMIN            = 63.15
        PGMAX           = 1.0E6
        TGMAX           = 120.0
AT,DOME,                63.15,      0.012530E6, 64.739E3+150.45E3
                        80.0,       0.13699E6,  79.065E3+116.02E3
C ELSE:                 90.0,       0.36071E6,  84.982E3+94.914E3
```

COMPLETE FPROP DATA BLOCK FOR NITROGEN:

```
HEADER FPROP DATA,7728,SI,0.0
C
C MORE COMPLETE N2 TWO-PHASE (FROM 63 TO 104K, NEAR 80 K)
C VAPOR PROPERTIES ARE FOR SAT VAPOR
C (EXCEPT CP WHICH MUST BE FOR LOW PRESSURE GAS)
C
        RGAS            = 8314.34/28.01
        TCRIT           = 126.2
        PCRIT           = 3.4E6
        TREFL           = 80.0,                 WSRK        = 0.0358
        ST              = 8.27E-3,              STTC        = 0.0 $ UNKNOWN STTC
        TMIN            = 63.15
        CPG             = 1.039E3,              CPGTC       = 0.0
        PGMAX           = 1.0E6
        TGMAX           = 125.0
AT,VG,  65.0,4.40E-6
        77.36,5.44E-6, 80.0,5.60E-6,            85.0,5.96E-6,           90.0,6.36E-6
        95.0,6.80E-6,  100.0,7.28E-6,           105.0,7.82E-6,          110.0,8.42E-6
        115.0,9.25E-6, 120.0,10.7E-6,           125.0,14.4E-6,          126.2,19.1E-6
AT,KG,  65.0,6.1E-3,    75.0,7.1E-3,            77.36,7.4E-3,           80.0,7.7E-3
        85.0,8.4E-3,    90.0,9.1E-3,            95.0,10.0E-3,           100.0,11.1E-3
        105.0,12.3E-3, 110.0,13.8E-3,           115.0,16.0E-3,          125.0,19.5E-3
C *** ALTERNATE INPUTS FOR VAPOR PROPERTIES ***************************************
C IF RUNNING ANALYSES FOR HIGH TEMP GAS, PRESSURES NEAR 1 ATM,
C THEN USE THE FOLLOWING LINES IN PLACE OF THE ABOVE 10 LINES:
C       PGMAX           = 2.0E5
C       TGMAX           = 500.0
CAT,VG, 65.0,4.40E-6
C       77.36,5.44E-6, 80.0,5.59E-6,            85.0,5.9E-6,            90.0,6.22E-6
C       95.0,6.54E-6,  100.0,6.87E-6,           105.0,7.19E-6,          110.0,7.52E-6
C       115.0,7.83E-6, 120.0,8.15E-6,           125.0,8.0E-6,           126.2,8.65E-6
C       130.0,8.78E-6, 140.0,9.4E-6,            180.0,11.8E-6,          200.0,12.9E-6,
C       220.0,13.9E-6, 240.0,15.0E-6,           260.0,16.0E-6,          280.0,16.9E-6,
C       300.0,17.9E-6, 340.0,19.7E-6,           440.0,23.7E-6,          460.0,24.4E-6,
C       480.0,25.2E-6, 500.0,25.9E-6
CAT,KG, 65.0,6.1E-3,    75.0,7.1E-3,            77.36,7.4E-3,           80.0,7.6E-3,
C       85.0,8.0E-3,    90.0,8.5E-3,            95.0,8.9E-3,            100.0,9.4E-3,
C       105.0,9.8E-3,  110.0,10.3E-3,           115.0,10.7E-3,          125.0,11.7E-3,
C       130.0,12.1E-3, 150.0,13.9E-3,           160.0,14.7E-3,          180.0,16.5E-3,
C       200.0,18.3E-3, 220.0,19.9E-3,           240.0,21.5E-3,          300.0,26.0E-3
C       320.0,27.4E-3, 340.0,28.7E-3,           380.0,31.3E-3,          400.0,32.5E-3,
C       480.0,37.5E-3, 500.0,38.6E-3
CAT,CPG,65.0,1.039E3,  320.0,1.039E3,           380.0,1.042E3,          460.0,1.050E3
C       500.0,1.056E3
C *** END ALTERNATE INPUTS ********************************************************
AT,DOME,                63.15,      0.012530E6, 64.739E3+150.45E3
                        80.0,       0.13699E6,  79.065E3+116.02E3
C ELSE:                 90.0,       0.36071E6,  84.982E3+94.914E3
AT,VL,  65.0,274.0E-6, 70.0,217.0E-6,           75.0,177.0E-6,          80.0,148.0E-6,
        85.0,127.0E-6, 90.0,110.0E-6,           95.0,97.2E-6,           100.0,86.9E-6,
        105.0,78.5E-6, 110.0,70.8E-6,           115.0,59.9E-6,          120.0,48.4E-6
        125.0,31.6E-6, 126.2,19.1E-6
AT,KL,  65.0,0.160,     70.0,0.151,             75.0,0.1413,            80.0,0.1322
        85.0,0.1231,    90.0,0.1142,            95.0,0.1053,            100.0,0.0966
        105.0,0.0880,  110.0,0.0795,            115.0,0.0710,           120.0,0.0628
AT,DL,  63.15,867.78,   65.0,860.78,            70.0,840.77,            75.0,819.22,
        80.0,796.24,    85.0,771.87,            90.0,745.99,            95.0,718.38,
        100.0,688.65,  105.0,656.20,            110.0,620.04,           115.0,578.14,
        120.0,525.12,  123.0,480.11,            125.0,431.03,           126.2,314.0
```

3.13.6 Advanced Two-Phase Fluid (6000 Series)

3.13.6.1 Introduction

This option allows the user to provide alternate fluid property routines. Fluids described with this method must be numbered between 6000 and 6999.

Unlike previous FPROP DATA options, this option requires the user to input *functional descriptions* of properties rather than *data values*. (Of course, the term "functional" does not preclude a function that merely looks up data values from a table.) The FPROP DATA block outlines only the *interface* between FLUINT and the user's routines; the user-supplied routines themselves may be provided in SUBROUTINE DATA or in an external user object library that is linked along with the processor library. Used in this manner, a 6NNN FPROP block behaves more like a logic block than a data block.

When using this option, the user must provide logic to replace about 10 FLUINT property routines, and may optionally replace any or all of the remaining set. To replace a routine, the user inputs Fortran statements in a subblock named after the routine that logic replaces (see Section 6.9.2). For example, if the keyword VPS is encountered in column 1, the statements that follow will be used to calculate saturation pressure P given temperature T. The Fortran logic is copied verbatim into a subroutine that is compiled and called from the processor; the logic can be arbitrarily complex and can call any other functions or subroutines supplied by the user. The user must respect the naming conventions used to interface their code with FLUINT (T=temperature, P=pressure, etc. See below). All subblocks for each fluid are placed in a single subroutine; a special subblock named VINIT may be used to input any initializations, declarations, and common blocks.

This property input option is considered advanced because the burden is placed on the user to produce a consistent and fast description of a fluid over a suitable property range. Again, this is actually a logic block disguised as a data block; the same care and debugging that applies to FLOGIC and VARIABLES blocks should be applied here. The use of the routine PRPMAP should be considered mandatory during the development of a 6NNN fluid description. The next subsection describes the speed and consistency requirements, and range limits are discussed in the subsequent subsection.

Fortunately, many property routine libraries are available that meet the requirements outlined below; the user need only be concerned with establishing the interface to such routines. Also, like any other FPROP DATA subblock, once a description has been developed it may be reused in any model as long as the referenced property routines are available in SUBROUTINE DATA or in an object library.

Although several such links to existing property databases have been made, difficulties have been encountered. Principally, execution speeds are disappointing since most such databases are designed for interactive look–ups and have not been streamlined for thousands of repeated calls. Other problems, such as different range limits and tolerancing schemes cause such problems as the return of a liquid property when a vapor property was requested. Many of these problems have been circumvented by the use of tabular look–ups, whereby the property databases are used to create large tables of data that are then interpolated by routines supplied in the 6NNN FPROP blocks. Examples of such tables, which have been used extensively for the analysis of real gases and supercritical cryogenic fluids, are available as templates.

3.13.6.2 Fluid Description Requirements

The property calculations should be thermodynamically consistent—the same state should be found as a function of any permutation of valid independent properties. For example, if an enthalpy H is found given temperature T and pressure P, then the same temperature should result when calculated using P and H. Also, performing a cyclic operation on a control volume should bring the state back to its original point. For these reasons, and because property derivatives are evaluated internally, the user exercise caution when employing curve fits to tabulated thermodynamic data, which may cause numerical problems. Methods for calculating self-consistent sets of properties are beyond the scope of this document. However, a good reference on this subject may be found in Section 2 of *Thermodynamic Properties in SI,* by W. C. Reynolds of the Stanford University Department of Mechanical Engineering.

The speed requirements for executed user code are critical. Unlike FLOGIC and VARIABLES blocks, which are only called a few times, property calculations are called *very* frequently. It is not uncommon to find routines like VH, VPS, and VSV called tens of thousands of times, even in relatively short and simple runs. In fact, *property calculation modules take by far more CPU time than do any other module in FLUINT, even matrix inversion modules!* The user must be therefore especially concerned with the execution speed of the subroutines they supply. This requirement usually precludes the use of generic integration routines; integration should be done by closed form solutions specific to each P-v-T surface. The use of divisions, logarithms, and powers should be minimized, and all iteration techniques must be as honed as possible.

When correctly done, tabular descriptions can be both fast and complete, although somewhat large. Unlike any other fluid description, their accuracy and speed are not very sensitive to the size of the domain. Unfortunately, generating such descriptions is not trivial nor easily described. Rather than attempt to document the algorithms and procedures required to duplicate such descriptions, the user is advised that tabular descriptions of water, ammonia, hydrogen, nitrogen, oxygen, ethane, and argon are available and should be used as templates for other fluids. The water and ammonia descriptions execute approximately 2 to 2.5 times faster than the corresponding standard library descriptions. Unless the fluid model is very small, tabular descriptions are the definitive choice for interfaces to large property libraries.

3.13.6.3 Regimes and Range Limits

Range limits are user-provided, except that temperatures must be within 10^{-3} to 10^{10}, and pressures must be less than 10^{10}. While the program will help enforce user–provided limits, *it is the user's responsibility to make sure that all properties are valid over the selected range, and that any necessary error trapping and handling logic is in place.* Unlike other user-described fluids, no additional checks will be made on the calculations provided. A error/warning message routine, UDFERR, has been provided to allow the user to write to the output file in a controlled manner. Note that slight tolerancing is used to avoid the limits, which are assumed to define the valid range exclusively (i.e., the input limits themselves represent invalid states).

This is the only fluid type that is allowed to operate above both supercritical temperature and pressure (if the user allows). By default, the fluid will be treated like a gas (XL=1.0) at any pressure if the temperature is supercritical. This requires a careful P-v-T and

vapor enthalpy calculation near the critical point to match the vapor regime with the liquid regime. If the valid region includes the critical point, all appropriate property routines should function at that point. This is important because the program immediately calculates and stores critical densities and enthalpies for repeated use thereafter in the program.

Figure 3–18 is provided as an illustration of the property range limits for two hypothetical fluids. For consistency, note that TGMAX is greater than or equal to TLMAX, and PGMAX is equal to VPS(TLMAX) if TLMAX is less than TCRIT.

Note that FLUINT assumes the liquid phase ($P > P_{sat}$ and $T < T_{crit}$) to be incompressible (XL=0.0). An incompressible formulation that is extrapolated from saturated liquid properties is used for liquid states with temperatures less than critical *at any pressure*, but a compressible (XL > 0.0) formulation is used within the dome and to the right of the isotherm containing the critical point. Supercritical transitions from compressible to incompressible (i.e., crossing the aforementioned isotherm) should be avoided without a carefully posed property description. For the "spongy" supercritical regime, the user can choose to employ the NEVERLIQ option (described below) to prevent a quality of 0.0 where an incompressible assumption is inappropriate.

NEVERLIQ Option

To avoid the compressible/incompressible discontinuity in the cold supercritical regime ($P > P_{crit}$ and $T < T_{crit}$), the user can choose to have the fluid treated as always compressible (XL=1.0). When this option is invoked, liquid, dome, and two–phase properties become optional.

There are two uses of this option. The first use is to apply this option to an existing full–range two–phase fluid description to avoid the aforementioned discontinuity by eliminating the possibility of a incompressible liquid phase. Vapor routines must then be able to return valid data over the entire range, and that range will generally be restricted to the supercritical zone ($P > P_{crit}$). The NEVERLIQ option is intended to be easily turned on and off to accommodate such usage.

The second use is for real gases, where liquid properties and domes are irrelevant. The primary purpose for using the NEVERLIQ option for such analyses is to enable to the user to supply a subset of the properties that would otherwise be required.

3.13.6.4 Input Formats

With the exception of a few data points such as the critical point and range limits, almost all information is supplied in the form of a subroutine or function call that is written directly into a Fortran subroutine by the preprocessor. As part of this interface, standard variable names are reserved (in these types of FPROP blocks only) as follows:

T ..... Temperature, local units as selected on the HEADER card
P ..... Pressure, *absolute* local units; *double precision*
H ..... Enthalpy, local units
S..... Entropy, local units
D ..... Density, local units
V ..... Specific volume, local units
CP ..... Specific heat, local units
U ..... Internal energy, local units
HFG .... Heat of vaporization, local units
X ..... Quality (vapor mass fraction)
A ..... Void fraction (vapor volume fraction)
VISC .... Viscosity, local units
COND... Conductivity, local units
SOS ..... Speed of sound, local units
METH .. Two–phase speed of sound method: 1 or 2; *integer*
ST ..... Surface tension, local units
DPDT ... Derivative of saturation pressure w.r.t. temperature (dP/dT)sat
DPARG .. RESERVED
SPARG .. RESERVED
INTARG . RESERVED

To be consistent with the remainder of the program, *pressures (P) must all be double precision*. The user should use the Fortran SNGL and DBLE functions and temporary variables to convert arguments for routines that treat pressures as single precision. The saturation pressure function VPS should return a double precision value. Similarly, the user may perform any internal unit conversions in the subblocks, as long as the incoming and outgoing values are in the unit set named on the HEADER card.

Calculations and subroutine calls may be placed within appropriate subblocks named after the FLUINT subroutine they will override (see Section 6.9.2). All such "V-style" subblocks are called functional subblocks. As with AT subblocks, these must occur after any keyword-style inputs. All FLUINT property routines may be overridden in this manner; but most can be are derived from the basic inputs provided by the user. For this reason, some routines are required whereas others are optional.

Information and code entered by the user in the subblocks are written verbatim to a central Fortran subroutine for each fluid (named UF6nnn, where 6nnn is the name input on the FPROP card). For example, to provide the thermal conductivity of the liquid phase, the user would enter Fortran instructions under the subblock named VCONDF, where the "V" starts in column 1 and denotes the start of a new subblock. Subblocks may be input in any order (with the exception of the special VINIT subblock described below, which must occur first if it is to be used).

Because all instructions and calls will be written to a single routine, it is the user's responsibility to avoid collisions between variables.

Calculations and subroutine calls should be performed in the set of units selected on the HEADER FPROP DATA card. (Pressures must be in absolute units.) FLUINT will provide the necessary unit conversions between these calculations and the unit system of the current master model. The user must make sure the property routines return units consistent with one of the sets listed in Appendix D.

A special block is provided that will be placed at the start of UF6nnn. This block, named VINIT, may be used to initialize or reset any parameters or common block variables needed for the fluid in question. This block is inserted between the UF6nnn declarations and the executable statements, and so can include both types of statements if desired, including DATA statements. Data common to all calculations may be placed or initialized here, or calls may be made here to a user initialization routine. Any executable statements place here will be executed *every time a property calculation for fluid 6nnn is made.* For this reason, any one-time initializations should be controlled by logic to avoid unnecessary repetition.

HEADER SUBBLOCK FORMATS:

```
HEADER FPROP DATA, 6nnn, [uid [,abszro] ]
     TCRIT=R, PCRIT=R
     [TMIN=R] [,TGMAX=R] [,PGMAX=R]
     [NEVERLIQ]
```

where:

TCRIT .... critical temperature. (No default. Not required if NEVERLIQ is invoked.)

PCRIT .... critical pressure, or minimum pressure if NEVERLIQ option is invoked. (No default.)

TMIN ..... lowest allowed temperature. (Lowest allowed pressure is the saturation pressure at this value or 10^{-3} minimum.)

TGMAX .... highest allowed gas temperature or 10^{10} maximum.

PGMAX .... highest allowed gas pressure. (Highest allowed liquid temperature is the minimum of the saturation temperature at this value or TCRIT or TGMAX or 10^{10} maximum.)

NEVERLIQ . if this keyword is present, signals the global use of the compressible assumption (quality will always be 1.0). PCRIT becomes the minimum pressure, vapor routines must function over the entire range (TMIN to TGMAX, PCRIT to PGMAX), and liquid and dome inputs are not required and will be processed if present, but ignored otherwise.

OPTIONAL FIRST FUNCTIONAL SUBBLOCK, Declarations and Initializations:

VINIT ..... Inserted in routine UF6nnn between declaration statements and first executable statements. May contain array dimensioning, common blocks, DATA statements, and executable lines.

REQUIRED FUNCTIONAL SUBBLOCKS (always):

VSV ..... Calculates: vapor specific volume V
Given: pressure P and temperature T
VH ..... Calculates: vapor enthalpy H
Given: pressure P and temperature T and vapor sp. volume V
VVISCV ..... Calculates: vapor dynamic viscosity VISC
Given: pressure P and temperature T
VCONDV .... Calculates: vapor conductivity COND
Given: pressure P and temperature T

REQUIRED FUNCTIONAL SUBBLOCKS (If NEVERLIQ not employed):

VPS ..... Calculates: saturation pressure P
Given: temperature T
VDPDTT .... Calculates: saturation derivative dP/dT
Given: temperature T.
(Must correspond to VPS values.)
VDL ..... Calculates: saturated liquid density D
Given: temperature T
VVISCF ..... Calculates: liquid dynamic viscosity VISC
Given: temperature T
VCONDF .... Calculates: liquid conductivity COND
Given: temperature T
VST ..... Calculates: liquid/vapor surface tension ST
Given: temperature T

OPTIONAL FUNCTIONAL SUBBLOCKS, Additional properties:

VS ..... Calculates: vapor entropy S
Given: temperature T and vapor sp. volume V
VSOS ..... Calculates: vapor speed of sound SOS
Given: pressure P and temperature T
(optional if VS provided)
VTAV1 ..... Calculates: vapor specific volume V and temperature T
Given: pressure P and entropy S
(Must correspond to VS and VSV values. Optional if VS provided)
VQUALS .... Calculates: quality X
Given: pressure P and temperature T
(optional if VS provided but recommended nonetheless)
VDLC ..... Calculates: compressed liquid density D
Given: pressure P and temperature T.
Must correspond to VDL at saturation.

OPTIONAL FUNCTIONAL SUBBLOCKS, More appropriate methods:

VTAV2 ..... Calculates: vapor specific volume V and temperature T
Given: pressure P and enthalpy H
(Must correspond to VH and VSV values.)

VTS ..... Calculates: saturation temperature T
Given: pressure P.
(Must correspond to VPS values, ***except input pressures exceeding the allowable saturation range should result in returned temperatures that are at those limits without exceeding them***.)

VDPDT ..... Calculates: saturation derivative DPDT
Given: pressure P.
(Must correspond to VPS and VDPDTT values.)

VCPV ..... Calculates: Vapor specific heat CP
Given: pressure P and temperature T.
(Must correspond to VH values.)

VCPF ..... Calculates: Liquid specific heat CP
Given: pressure P and temperature T.
(Must correspond to VH, VHFG values.)

VHLIQ ..... Calculates: Liquid enthalpy H
Given: pressure P and temperature T.
(Must correspond to VH, VHFG values.)

VTLIQ ..... Calculates: Liquid temperature T
Given: pressure P and enthalpy H.
(Must correspond to VH and VHFG values.)

VULIQ ..... Liquid internal energy U
Given: pressure P and temperature T.
(Must correspond to VH, VHFG values.)

VQUALH .... Calculates: Quality X
Given: pressure P and enthalpy H.
(Must correspond to VH and VHFG values.)

VHFG ..... Calculated: Heat of vaporization HFG
Given: pressure P and temperature T and vapor sp. volume V
(Must correspond to VH and VHLIQ values.)

VALPHA .... Calculates: Void fraction A
Given: temperature T and quality X.
(Must correspond to VSV and VDL.)

VSOSF ..... Calculates: liquid speed of sound SOS
Given: pressure P and temperature T
(optional if VDLC provided)

VSOSFV .... Calculates: liquid, vapor, or two–phase speed of sound SOS
Given: pressure P, temperature T, quality X, and method METH
(optional if either VDLC or VSOSF *and* either VSOS or VS provided)

3.13.6.5 Example 6000 Series FPROP DATA Block

In the following block, a description of ammonia is input using current internal FLUINT property routines as externally provided routines, such that no SUBROUTINE DATA or additional processor libraries are needed. While there is obviously no reason to input such a description when one already exists (as fluid 717), this simple example should illustrate this option using a set of property routines already available to all users. Refer also to Section 7.3 (RAPPR) for a computer generated example of a 6000 series block.

In these routines, a pointer to a data array must be initialized in the VINIT subblock. Also, the value of a logical flag SI (held in common RCONV) must be set to false before each call, and then reset to its previous value. These are examples of the types of operations that might be performed in typical VINIT blocks.

Note that the bare minimum of routines is input. Entropy and entropy-based calculations will not be available for this fluid, and standard calculation methods will be used to derive all other necessary properties. If the user had better, faster, or otherwise more appropriate property routines, these could have been added to the following list.

```
HEADER FPROP DATA,6717,ENG,0.0
C
C AMMONIA IN ENGLISH UNITS USING STANDARD LIBRARY ROUTINES
C !!!!!! EXAMPLE ONLY !!!!!! SEE ALSO DESCRIPTIONS GENERATED BY RAPPR.
C
        TGMAX           = 3.0*730.0799,         PGMAX       = 1636.37
        TCRIT           = 730.0799,             PCRIT       = 1636.37
        TMIN            = 351.8
VINIT
        DOUBLE PRECISION ZPS
        COMMON/FLUDAT/IFI(90),FD(1460)
        COMMON /RCONV/ PCON,TCON,HCON,VCON,CCON,SI,PVC,PTOWD
        LOGICAL SI,SIL
        N               = 372
        SIL             = SI
        SI              = .FALSE.
VPS
        P               = ZPS(T,FD(N))
        SI              = SIL
VDPDTT
        DPDT            = ZDPDTT(T,FD(N))
        SI              = SIL
VH
        H               = ZH(P,T,V,FD(N))
        SI              = SIL
VSV
        V               = ZSV(P,T,FD(N))
        SI              = SIL
VST
        ST              = ZST(T,FD(N))
        SI              = SIL
VCONDF
        COND            = ZCONDF(T,FD(N))
        SI              = SIL
VCONDV
        COND            = ZCONDV(T,FD(N))
        SI              = SIL
VVISCF
        VISC            = ZVISCF(T,FD(N))
        SI              = SIL
VVISCV
        VISC            = ZVISCV(T,FD(N))
        SI              = SIL
VDL
        D               = ZDL(T,FD(N))
        SI              = SIL
```

The listing on the next page is the Fortran subroutine produced by FLUINT based on the above inputs. Although the user normally does not need to look at the resulting subroutine, it is presented here to help illustrate concepts.

In the master model in which this input file was run, the unit system was SI, and ABZSRO=0.0. Note the unit conversions performed before and after each subblock. The encoding and decoding of the DPARG, SPARG, and INTARG arrays are handled by the program.

```
      SUBROUTINE UF6717(DPARG,SPARG,INTARG)
      DOUBLE PRECISION DPARG(1),P
      DIMENSION SPARG(1),INTARG(1)
C VINIT
      DOUBLE PRECISION ZPS
      COMMON/FLUDAT/IFI(90),FD(1460)
      COMMON /RCONV/ PCON,TCON,HCON,VCON,CCON,SI,PVC,PTOWD
      LOGICAL SI,SIL
      N              = 372
      SIL            = SI
      SI             = .FALSE.
C VPS
      IF    (INTARG(1) .EQ.    1) THEN
      T = SPARG(1) *   1.8000000
      P              = ZPS(T,FD(N))
      SI             = SIL
      DPARG(1) = P *  6.89475977d+03
C VDPDTT
      ELSEIF(INTARG(1) .EQ.    2) THEN
      T = SPARG(1) *   1.8000000
      DPDT           = ZDPDTT(T,FD(N))
      SI             = SIL
      SPARG(   2) = DPDT *   12410.567
C VH
      ELSEIF(INTARG(1) .EQ.    3) THEN
      T = SPARG(1) *   1.8000000
      P = DPARG(1) *  1.45037688d-04
      V = SPARG(2) *   16.018463
      H              = ZH(P,T,V,FD(N))
      SI             = SIL
      SPARG(   3) = H *   2326.1113
C VSV
      ELSEIF(INTARG(1) .EQ.    4) THEN
      T = SPARG(1) *   1.8000000
      P = DPARG(1) *  1.45037688d-04
      V              = ZSV(P,T,FD(N))
      SI             = SIL
      SPARG(   2) = V *  6.24279603e-02
C VST
      ELSEIF(INTARG(1) .EQ.    5) THEN
      T = SPARG(1) *   1.8000000
      ST             = ZST(T,FD(N))
      SI             = SIL
      SPARG(   2) = ST *   14.593176
C VCONDF
      ELSEIF(INTARG(1) .EQ.    6) THEN
      T = SPARG(1) *   1.8000000
      P = DPARG(1) *  1.45037688d-04
      COND           = ZCONDF(T,FD(N))
      SI             = SIL
      SPARG(   2) = COND *   1.7309999
C VCONDV
      ELSEIF(INTARG(1) .EQ.    7) THEN
      T = SPARG(1) *   1.8000000
      P = DPARG(1) *  1.45037688d-04
      COND           = ZCONDV(T,FD(N))
      SI             = SIL
      SPARG(   2) = COND *   1.7309999
C VVISCF
      ELSEIF(INTARG(1) .EQ.    8) THEN
      T = SPARG(1) *   1.8000000
```

```
        P = DPARG(1) *  1.45037688d-04
        VISC           = ZVISCF(T,FD(N))
        SI             = SIL
        SPARG(   2) = VISC *  4.13333328e-04
C VVISCV
       ELSEIF(INTARG(1) .EQ.    9) THEN
        T = SPARG(1) *   1.8000000
        P = DPARG(1) *  1.45037688d-04
        VISC           = ZVISCV(T,FD(N))
        SI             = SIL
        SPARG(   2) = VISC *  4.13333328e-04
C VDL
       ELSEIF(INTARG(1) .EQ.   10) THEN
        T = SPARG(1) *   1.8000000
        D              = ZDL(T,FD(N))
        SI             = SIL
        SPARG(   2) = D *   16.018463
C
       ELSE
         CALL VERROR('UF6717','NO SUCH PROPERTY',6717)
       ENDIF
       RETURN
       END
```

4. USER GUIDANCE AND SPECIALIZED OPERATIONS

4.1 Introduction

This section contains material that does not logically fit into a presentation of input option and features. The information presented here in the style of a tutorial should prove very useful to the new user once he has achieved some familiarity with the input options and requirements. The experienced user should find this section useful for reference and review.

Section 4.2 contains a summary of various Fortran statements that are available to control program flow in the processor. Users completely unfamiliar with Fortran should use this section as a supplement to a basic Fortran text.

Section 4.3 explains the mechanics of making parametric runs and restarting from previous runs. There is almost no commonality between SINDA/FLUINT and previous versions of SINDA in this arena, so this section is recommended reading for all users.

Section 4.4 is a tutorial in the use of the INCLUDE type header record which allows users with large models to manage their input data in a way that avoids any need to assemble the input data into a single, cumbersome entity until run time. This section is recommended reading for all users.

Section 4.5 is an introduction to the SINDA/FLUINT library. The conventions used in Section 6, where each user-callable subroutine is described, are presented in this section along with a short overview that gives the new user an idea of the capabilities of the processor library.

Section 4.6 is a discussion of the program variables that contain the status of the problem as the solution proceeds, and the program control "constants" that enable the user to alter the course of the solution as desired. This section should be valuable to the new users as a tutorial in program control. Experienced users will find it useful as reference material.

Section 4.7 is an introduction to fluid system modeling using FLUINT. This section is intended to be read before attempting to generate a fluid model, and may be used as reference thereafter. This section describes fluid networks and their basic components, defines major program variables, input methods, modeling techniques, and trouble shooting.

4.2 Fortran Control Statements

The purpose of this section is to familiarize the user with the basic types of Fortran statements which are available for implementing program flow control. No attempt will be made to instruct the user in the formal details of utilizing such statements correctly, since such details may be found in any Fortran text*. The following list shows the types of Fortran statements which may be used to effect program control and gives an example of each:

Statement Type		Example
		`1 7`
1. CONTINUE		`F 450 CONTINUE`
2. unconditional GO TO		`F GO TO 350`
3. computed GO TO		`F 160 GO TO (200,250,300) M`
4. assigned GO TO		`F GO TO N`
5. arithmetic IF		`F 200 IF (ITEST+KTEST) 250,300,400`
6. logical IF		`F IF (RTEST.LE.STEST) GOTO 2`
7. DO loop	F	`DO 450 M=1,3`

The CONTINUE statement causes no action to occur, but is useful as a dummy statement to which a statement number may be assigned. CONTINUE statements are often used to set up a skeleton framework of statement numbers for program control.

The unconditional GO TO statement is used to transfer control directly to the statement prefixed by a certain statement number. This statement number must be known and specified at compile-time and may not be changed at run-time.

The computed GO TO statement permits control to be transferred to the statement prefixed by the Nth statement number in a list of statement numbers, where N is supplied as an integer variable.

The assigned GO TO statement is similar to the unconditional GO TO except that the statement number to which control will be transferred is specified as a variable. This variable is associated with a specific statement number by using an ASSIGN statement.

The arithmetic IF statement is used to transfer control to one of three specified statement numbers, depending on whether a given arithmetic expression is negative, zero, or positive.

* See for instance, J.W. Perry Cole, *ANSI Fortran IV with Fortran 77 Extensions*, Second Edition Wm. C. Brown Co., Dubuque, Iowa, 1983.

The logical IF statement is used to cause the execution of a given statement if, and only if, a logical expression is true. Logical operators include .NOT., .OR., and .AND. and may operate on logical expressions or variables. Arithmetic expressions connected by relational operators are logical expressions. The following relational operators may be used:

.EQ.	equal to
.NE.	not equal to
.LT.	less than
.GT.	greater than
.LE.	less than or equal to
.GE.	greater than or equal to

The DO loop may be used to cause the execution of a group of statements to be repeated for successively incremented values of a so-called "index variable."

The statements used as examples thus far in this section are true Fortran statements, and the F in column 1 is thus appropriate. The user should be aware, however that M-type Fortran statements may also be employed for program control. For instance, a variant of Example 3 might be:

```
1      7
160    GO TO (200,250,300) YSIDE.K180
```

Here, the simple variable M is replaced by user constant 180 in submodel YSIDE. Since an F does not appear in column 1, the preprocessor will translate the user constant reference into an internal constant number (e.g. 3) and re-write the statement as follows:

```
160    GO TO (200, 250, 300), K(3)
```

Note that the translation only applied to the user constant reference. The remainder of the statement had to conform to Fortran syntax.

If the user required user constant references in Example 4 and 6, M-type statements could be used, that is:

```
7
GO TO G.K12
IF(RTEST.LE.G.K100) ASSIGN 160 TO G.K12
```

The DO loop in Example 7, could also reference a user constant, that is:

```
7
DO   450 M = 1, YSIDE.K140
```

4.3 Multiple Case/Multiple Run Operations

The SINDA/FLUINT system includes the capability to perform parametric analyses during a single run as well as restarting subsequent runs from saved data. These two capabilities are independent of each other. Parameter runs can be made with or without restart and a run may be restarted with or without the parameter options being used. The one restriction that applies to both of these capabilities is that neither works on a per submodel basis. The entire set of active and inactive submodels and data will be both saved and retrieved.

With regard to restarts, *the structure of the network must be the same when the data is retrieved as it was when it was saved*. The values of any network parameters may be changed, but no network elements, user constants, arrays etc. may be added or deleted, and the input order cannot change. Otherwise, a fatal error will be detected, with a message detailing the reason for the collision. However, collision checks cannot detect changes in input order. (An alternate restart routine is available to circumvent such collision checks if the user wishes to assume responsibility for potentially catastrophic failures in exchange for greater modeling power.) Parametric capabilities, of course, are not affected by this restriction because they are performed within a single run.

4.3.1 Multiple Cases Within a Single Run

Frequently, a user will want to run multiple cases of the same basic thermal or flow problem to see, for example, how his model responds to changes in boundary conditions. This is generally an efficient way to operate because preprocessing, compiling and loading operations are not repeated. This parametric run capability is accomplished using subroutines SAVPAR & RESPAR.* SAVPAR represents a means of taking a "snapshot" of a model and its status at any time within the processor execution. This snapshot may then be restored for further analyses through the use of RESPAR. This is known as a parametric analysis procedure.

When SAVPAR is called it writes out all the current values of the T, C, Q, G arrays, all fluid submodel data, and the values of the control constants, user constants (numbered, but not named) and user arrays. Each call to SAVPAR will return a unique record number. These record numbers serve as identifiers for the snapshots, and should be stored in user constants or in a user array for subsequent use in subroutine RESPAR, which requires that record number as an argument. Incorrect arguments will either result in a collision error message or restoration of the wrong snapshot. After a RESPAR call, all the data in the processor has been restored to the values it had when the corresponding SAVPAR was called.

SAVPAR writes out a lot of data. For this reason it should probably never be called from a VARIABLES or FLOGIC block. SAVPAR is most suited for calls from OPERATIONS DATA or OUTPUT CALLS. It should only be called from a VARIABLES or FLOGIC block if the number of calls are restricted by user logic.

One possible mode of parametric operation is as follows:

1) save data: - CALL SAVPAR(NTEST)

* These subroutines replace the less general Initial and Final Parameters options used with previous versions of SINDA.

2) execute solution routine: - CALL FWDBCK

3) restore data: - CALL RESPAR(NTEST)

4) change data values as required: - RTG.C10 = RTG.C10*1.1
- etc.

5) repeat 1 through 4 as required

4.3.2 Restart Operations

Restart operations involve saving the data values from one run and then reusing them in another run. For example, one run could perform a one hour simulation and save the appropriate data values. Another run could read in these values and perform the second hour of the simulation. Alternatively, steady-state results could be saved as for use as initial conditions in different transient analyses.

The restart capabilities are invoked through two routines: RESAVE and RESTAR. To use restart one or both of the following lines must appear in OPTIONS DATA.

```
RSO  =  filename
RSI  =  filename
```

The RSO line must appear in order to create a restart file. The RSI line must appear in order to read data from a previously created restart file. Both lines must appear in order to read one restart file and create another in the same run.

The restart file (RSO) is written using RESAVE routine. This routine has one argument: a string composed of one or more of the following letters:

T ..... to save temperatures
C ..... to save capacitance
Q ..... to save Q values
G ..... to save conductor values
A ..... to save user arrays
U ..... to save user constants (numbered and named)
N ..... to save control constants
L ..... to save lump PL, TL, XL, and QDOT (for plotting)
P ..... to save path FR (for plotting)
F ..... to save remaining FLUINT data (*necessary for restarts*)
ALL ..... to save all of the above

Examples:

```
CALL RESAVE('ALL')
CALL RESAVE('TCQ')
CALL RESAVE('TCQGAUN')
```

When RESAVE is called it will write a record number to the output file with the current problem time, as illustrated by the following example:

```
RESTART RECORD NUMBER 1 WRITTEN FOR TIMEN = 1.75
```

RESAVE will print a different record number each time it is called. This record number must be used as an argument in any subsequent runs calling RESTAR. Calling RESTAR with the appropriate record number will restore the data that was saved at that time. If a number of different saves were made with RESAVE, a number of different restarts can be made by calling RESTAR with different arguments. The same restart can be repeated several times or a series or different restarts may be performed from the same RSI file. Incorrect RESTAR arguments will either result in a collision error message or restoration of the wrong snapshot.

The *advanced* user may elect to use RESTNC in place of RESTAR. RESTNC is identical to RESTAR except that no collision checks are performed, which can result in unpredictable behavior and potentially catastrophic errors if used improperly. The RESTNC user is assumed to have detailed knowledge of both SINDA/FLUINT storage rules and restart file operations.

RESAVE writes out a lot of data. For this reason RESAVE should probably never be called from VARIABLES or FLOGIC blocks. RESAVE is most suited for calls from OPERATIONS DATA or OUTPUT CALLS. It should only be called from a VARIABLES or FLOGIC block if the number of call are restricted by user input logic.

An example of the restart operation is as follows:

A) Run 1

1) perform solution routine	- CALL FWDBCK
2) save data in OPERATIONS or OUTPUT	- CALL RESAVE('ALL')

B) Run 2 (assume previous RESAVE wrote out record 1)

1) restore data	- CALL RESTAR(1)
2) change data	- TIMEND = 5.0
	- etc.
3) use data	- CALL FWDBCK

Caution—The actions of the routines DRPMOD and PUTTIE are not preserved from a previous run. In other words, RESTAR and RESTNC do not affect either the thermal submodel dormant/awake status nor the location of the ties.

Caution—Named user constants are never saved nor retrieved: they are intentionally invariant since they are intended for use in controlling restart and parametric operations.

4.3.3 Crash Files: Abort Recovery

Crash files are an alternative restart option that are particularly suited to helping the user recover from unexpected termination of the run without resorting to repeated calls to RESAVE tend to result in huge files.

Crash files are created and updated via calls to a single routine, CRASH, which is equivalent to calling RESAVE with an argument of 'ALL', *except that repeated calls overwrite the previously stored snapshot.* Therefore, CRASH can be called an arbitrary number of times without causing the crash file size to grow. Also, the contents (file buffers) are flushed each call, so that a subsequent failure or abort does not render the last saved state invalid.

The cost of a call to CRASH is not completely insignificant; the user must trade–off how often to call it versus how much ground must be retraced in the event of a crash. Calling it from FLOGIC 0 or VARIABLES 0 will maximize both its cost and its 'freshness,' while calling it from OPERATIONS DATA will minimize both. OUTPUT CALLS is probably the most logical compromise location for most uses. Often, not all OUTPUT CALLS blocks are used for all submodels; using such a otherwise empty block enables the user to customize the frequency of calls to CRASH.

The crash file is closed after each call, and is reopened with STATUS='UNKNOWN' on the next available unit number. The first call to CRASH will generate an output file message containing the record number: "CRASH FILE RECORD NUMBER NNN." This record number should be used in subsequent calls to RESTAR, with the crash file name input as the RSI file in OPTIONS DATA.

CRASH files are not named in OPTIONS DATA. The sole argument to the CRASH routine is the file name to be used, entered either in single quotes or as a CARRAY reference. On case–sensitive machines (e.g., Unix), use the CARRAY option to preserve lower case letters in file names. Otherwise, the file name will be capitalized.

Examples:

```
CALL CRASH('OHNO')      $ LOCAL FILE NAME
CALL CRASH(UCA10)       $ FILE NAME INPUT AS CARRAY 10
```

Caution—Calls to CRASH may appear in more than one place in the model *as long as the file names are identical.* Otherwise, fatal errors may result upon attempting to restart.

4.4 The Use of INCLUDE Directives

4.4.1 General Information

The INCLUDE directive fetches all or part of another file and places it in the current input file. The INCLUDE directive may be used in one of two forms; (1) it may act as substitute for a header record, or (2) it may be used to add information to an existing header. Header records may also be contained within an INCLUDE file, however, the caution at the end of this subsection should be noted in such cases.

The general form of the INCLUDE directive is as follows:

```
INCLUDE pfn [,ln1 [:ln2]] [,other options]
```

where:

`pfn`	a permanent file name
`lnx`	line number range in pfn to include (all of pfn by default)
`options` ..	as defined by the header card being substituted for
`[ ]`	indicates optional input

Two examples of the use of the INCLUDE directive are as follows. (Note that dollar sign style comments cannot be used with INCLUDEs). As an example of data being inserted into an existing header block, assume that all radiation conductors have been stored in a file called RADK.DAT. An INCLUDE directive inside of CONDUCTOR DATA would add the conductors described in RADK.DAT to the rest of the block.

```
C EXAMPLE 1
INCLUDE RADK.DAT
```

To show how an INCLUDE directive can replace a header block, assume that all conductors for a single submodel have been stored in a file called COND. The SINDA/FLUINT input file would then contain an INCLUDE record instead of a CONDUCTOR header. This INCLUDE directive would then fetch all of the conductor data contained in COND, prefacing it with the appropriate header instruction, as shown below:

```
C EXAMPLE 2
INCLUDE COND,,CONDUCTOR DATA, MODEL1
```

When the INCLUDE directive is being used as a substitute for a header record, then "INCLUDE pfn [,ln1 [:ln2]]" is a substitute for the word HEADER. The rest of the INCLUDE card options are simply the remaining portions of the header record in question. See Section 3.3.1 for more information on header records.

The INCLUDE directive can be used to fetch selected portions of a file. The input option "[,ln1 [:1n2]]" specifies the lines to be fetched. If no lines are specified the entire file is read (Example 3). If only one number is specified, it is assumed to mean the number of lines to fetch *starting at the beginning of the file* (Example 4). If both line numbers are specified (Example 5), they are assumed to define a range of lines to be fetched (e.g., fetch from line ln1 to ln2, inclusive).

```
C EXAMPLE 3
INCLUDE COND

C EXAMPLE 4
INCLUDE COND,10

C EXAMPLE 5
INCLUDE COND,10:20
```

If no lines are specified and the INCLUDE card is used as a substitute for a header record, then an empty set of commas must mark this position, as shown in Example 2. Note that if file COND actually began with a header record, the format in Example 2 would not be required, but would still be recommended because it makes the input file more self-documenting.

4.4.2 Suggested Modes of Operation

Probably the best mode of operation is to use the INCLUDE directive to bring information into existing headers. In this mode, large blocks of data that make the input file hard to handle can be kept separate while still keeping the structure of the input file clear to a reader.

An equally valid mode is the use of the INCLUDE directive as a substitute for one header record. In this mode, large blocks of data are also kept separate while still retaining essential information in the input file. If INCLUDE files contain headers, then it is not apparent from the input file which information is being included without further comments. For example, if the include directive INCLUDE COND,,CONDUCTOR DATA,MODEL1 appeared it would not be apparent that COND also contained HEADER CONTROL DATA for MODEL1.

Alternatively, the INCLUDE file can wholly contain one or more header blocks. This is the recommended mode for FPROP blocks.

Caution—INCLUDE directives function somewhat like Fortran subroutine calls; the program does not just blindly include all files and *then* begin to preprocess the entire input deck. Rather, upon returning from an INCLUDE operation, *the program assumes it is still inside the same header block it was in when the INCLUDE was encountered.* This may cause problems if HEADER records are contained within an INCLUDE file, but a new HEADER is not encountered immediately after returning from an INCLUDE operation.

Caution—INCLUDE directives cannot be nested (i.e., an INCLUDE file cannot contain another INCLUDE directive to a third file.

4.5 Introduction to the SINDA/FLUINT Library

4.5.1 Basic Conventions

The SINDA/FLUINT library is a collection of thermal and fluid network solution subroutines together with a set of support subroutines that perform the input/output, restart, parametric run and numerical interpolation, and other specialized simulations. The fluid flow subroutines (SINFLO) and some of the numerical manipulation subroutines that are in the old SINDA library have not been carried forward to SINDA/FLUINT, having been replaced by FLUINT.

The description and calling sequence for each subroutine is given in Section 6. However, introductory remarks and general usage recommendations are included in this section with the intent of giving the novice user some insight into the scope of the available routines, as well as identifying those routines and associated techniques which enjoy frequent usage in real engineering applications. Experienced users should also find these comments helpful in that they point out certain economies and techniques which were not possible with the old CINDA-3G system. Consider interpolation subroutine D1D1DA, whose calling sequence is as follows:

```
CALL D1D1DA (X,AX(IC),AY(IC),Y)
```

where:

X	Value of independent variable
AX	Singlet array reference of the form smn.An or An
AY	Singlet array reference to array of Y-values corresponding to each X-value in AX (form smn.An or An)
Y	Dependent variable result

It will be noted, first, that there are no record column designations. The word "CALL" begins to the right of column 6, in keeping with FORTRAN convention. Next, it will be seen that the descriptions of the arguments are rather terse and make no explicit mention of arithmetic type or permissible reference forms. Arithmetic type is always implied by the first letter of each formal dummy argument, in accordance with implicit Fortran conventions. Letters I through N, inclusive, imply integer arguments; all other letters imply floating point arguments. If an argument must be a character string, this fact will be stated explicitly. For example, D1D1DA requires two simple arguments, X and Y, which must be floating point.

Arguments which represent arrays are indicated by suffixing the formal argument with a parenthesized "IC" or "DV." IC indicates that the argument supplied by the user must be an array reference of the integer-count form. This terminology results from the fact the first cell of all SINDA/FLUINT arrays is equivalenced to an integer containing the array size, IC, followed by IC data values. For example, D1D1DA requires two arrays, AX and AY, which must contain floating point numbers and which must be supplied as references of the form: smn.An, where n is the array number and smn is the associated submodel name. DV indicates that the argument supplied by the user may be an array reference of the data-value form or any type of reference which is associated with a single data value or the starting location of a sequence of data values.

A floating point argument is expected—the use of the above form (smn.An) would cause an error even if array smn.An contained floating point values because the first cell is equivalenced to an integer value, and the subroutine is expecting a single floating point value.

When several similar subroutines are described, the formal description of similar arguments will be deleted. For example, consider subroutine D11DAI. This routine performs single variable linear interpolation on an array of X's to produce an array of Y's. The number of input X's must be supplied separately as the argument N and must agree with the number of output Y's. The calling sequence is as follows:

```
CALL D11DAI  (N,X(DV),AX(IC),AY(IC),Y(DV))
```

It should be clear, from the basic action of the routine and the description of the arguments for D1D1DA, that the required arguments for D11DAI are as follows:

N ..... Number of values in X and Y arrays (integer)
X ..... Array of input values of the independent variable
AX ..... Singlet array of X's in ascending order (integer count form)
AY ..... Singlet array of Y's corresponding to the X's in AX (IC form)
Y ..... Array of output values of the dependent variable

D11DAI is equivalent to N calls to D1D1DA. If these conventions are not obvious, the user should not become alarmed. Rather, he should recognize that ease and confidence in understanding and using the more complex routines in the library come only after experience is gained in using the simpler ones.

4.5.2 Execution Subroutines

These routines are called "execution routines" because they are called from the OPERATIONS DATA block. They operate on the network defined by the BUILD and BUILDF cards that are in effect and usually call VARIABLES 0, VARIABLES 1, VARIABLES 2, FLOGIC 0, FLOGIC 1, FLOGIC 2 and OUTPUT CALLS automatically. Thus, these routines form the core of most thermal type problem solutions. This section contains two classes of routines: (1) network solution routines, and (2) network mapping routines.

The network solution routines offer five basic combinations of algorithms for determining the transient or steady state solution of a thermal and/or fluid problem. These routines are: FASTIC, STDSTL, FWDBCK, FORWRD, and FWDMTS. FASTIC and STDSTL find the steady-state of the thermal and fluid networks (i.e., the conditions that would exist if the system were unperturbed for a long period of time). FWDBCK, FORWRD, and FWDMTS track the transient changes in the system as a function of time. Each routine is described in Section 6.2.

Note that FASTIC and STDSTL are *identical* with respect to thermal network solutions, and that FWDBCK and FORWRD are *identical* with respect to fluid network solutions. *FWDMTS has no affect on fluid networks.* This section describes only the solutions of the thermal networks; the methods used to find the simultaneous solutions for fluid networks are described in Appendix E.

The steady state routines STDSTL and FASTIC seeks the roots of the equation:

$$\mathbf{Q} = f(\mathbf{T})$$

where:

Q ..... Heat rates
T ..... Temperature of nodes

The basic equation used for all thermal transient analyses is as follows (neglecting impressed Q's):

$$\mathbf{T}_{new} = \mathbf{T}_{old} + f(\lambda * \mathbf{T}_{new}, (1-\lambda) * \mathbf{T}_{old})$$

where λ is a weighting factor: 0 .LE. λ .LE. 1

Subroutines FORWRD and FWDMTS used a formulation where the parameter $\lambda = 0$. This method is called first-order *explicit forward differencing*. In formulations where $\lambda \neq 0$, it will be seen that $\mathbf{T}_{new}$ appears on the left and right hand side of the equation. Clearly, these methods require a either a simultaneous or iterative solution. Hence, formulations with $\lambda \neq 0$ are called *implicit* solutions. This is the method used by FWDBCK, which is also called second-order implicit differencing, trapezoidal integration, or the Crank-Nicholson method

There are two network mapping routines: QMAP and FLOMAP. The thermal network and heat flow mapping routine, QMAP, serves as a means of converting the internal node sequence into an orderly presentation of the thermal network it represents. The QMAP output lists the capacitance, stability factor ($C/\Sigma G$, a minimum characteristic time), adjoining nodes and interconnecting conductor values for each node in the network. If meaningful temperatures are available when QMAP is called, a listing of the heat flow rates through each conductor is also obtained. The FLOMAP routine performs an analogous function for fluid models, and produces output in the same file as does QMAP. Two additional routines are available to produce mappings of individual nodes and lumps: NODMAP and LMPMAP.

4.5.3 Interpolation Subroutines

Section 6.4 describes a large variety of interpolation routines. Analytically minded users might, at first, shun the concept of approximating smooth curves with a series of straight lines. However, since SINDA is built around discretized variables, interpolation is the method of choice for evaluating time or temperature dependent properties and single or multi-variable functions. Though parabolic interpolation is also available, basic linear interpolation serves the great majority of user's needs:

$$Y = Y_1 + \frac{(Y_2 - Y_1)}{(X_2 - X_1)} (X - X_1)$$

Linear interpolation is available in two elementary forms: subroutine D1DEG1, which requires one doublet array of (Xi,Yi) ordered pairs, and subroutine D1D1DA, which requires two matching singlet arrays, one containing values of Xi and the other containing corresponding values of Yi. If the user must approximate several dependent variables over roughly the same domain of a single independent variable, then the use of subroutine D1D1DA, along with its associated array structure, will be preferred.

4.5.4 Input and Output Subroutines

Section 6.3 describes the routines the user might use to construct his OUTPUT CALLS block. This set of routines provide printed output as well as output to disk files. The disk file outputs are for use in parametric runs, restart runs and for external programs such as EXPLOT, which provides graphical output of time dependent data. Section 6.4 describes the routines used for reading data from disk files for use by the program. The restart and parametric output routines each have a counterpart input routine to retrieve the data. Other input routines are available for reading in time dependent boundary temperatures and heat source data that are written to disk files by external programs. Fluid submodel output routines are described in Section 6.9.5.

4.5.5 Application and Utility Subroutines

Section 6.6 and 6.7 describe the utility subroutines and application subroutines. Utility subroutines allow the user to conveniently locate such things as submodel names and other model attributes in the form of Fortran-addressable memory locations. Application subroutines are devoted to specific applications such as phase change and heat exchanger modeling.

FLUINT utility and application routines are described in Section 6.9.

4.6 Advanced Control Constant Usage

This section discusses control of thermal submodels only. Refer to Section 4.7.11 for control of fluid submodels. The various SINDA control constants may be divided into two operational groups, as follows:

1) Program-calculated control constants

2) User-specified control constants

Program-calculated control constants contain values output by the network solution routines as results of the solution routine calculations. They may be examined by the user at his discretion and may be used as criteria for controlling the operations in any of the user logic blocks. The following lists show the names of the program-calculated control constants.

Program-calculated single-value constants:

```
DATE,   LINECT,   LOOPCT,   MLINE,   MMODS,   NMACT,
NNOD,   PAGECT,   TIMDY,    TIMEM,   TIMEN,   TIMEO
```

Program-calculated multi-value constants:

```
smn.ARLXCC,   smn.ATMPCC,   smn.CSGMAX,   smn.CSGMIN,
smn.DRLXCC,   smn.DTIMEU,   smn.DTMPCC,   smn.EBALNC,
smn.EBALSC,   smn.ESUMIS,   smn.ESUMOS,   smn.NARLXC,
smn.NARLXN,   smn.NATMPC,   smn.NATMPN,   smn.NCGMAC,
smn.NCGMAN,   smn.NCSGMC,   smn.NCSGMN,   smn.NDRLXC,
smn.NDRLXN,   smn.NDTMPC,   smn.NDTMPN,   smn.NEBALC,
smn.NEBALN,   fsmn.DTIMEF
```

where:

`smn` ..... thermal submodel name
`fsmn` ..... fluid submodel name

User-specified control constants are used to influence the calculations that are internal to the network solution routines. The following lists show the names of the user-specified control constants.

User-specified single-value constants:

```
ABSZRO, DTIMES, NLOOPS, SIGMA, TIMEN, TIMEND, TIMEO, PATMOS,
ACCELX, ACCELY, ACCELZ
```

User-specified multi-value constants:

```
smn.ARLXCA,  smn.ATMPCA,  smn.BACKUP,  smn.CSGFAC,
smn.DRLXCA,  smn.DTIMEH,  smn.DTIMEI,  smn.DTIMEL,
smn.DTMPCA,  smn.EBALNA,  smn.EBALSA,  smn.EXTLIM,
smn.ITEROT,  smn.ITERXT,  smn.ITHOLD,  smn.NLOOPT,
smn.OPEITR,  smn.OUTPUT,  fsmn.DTSIZF, fsmn.DTMAXF,
fsmn.OUTPTF, fsmn.RSSIZF, fsmn.RSMAXF, fsmn.RERRF,
fsmn.REBALF, fsmn.ITROTF, fsmn.OPITRF, fsmn.DTMINF,
fsmn.ITHLDF, fsmn.DTTUBF, fsmn.RSTUBF
```

Internal control constants not explained in Section 3.5.2:

NDTMPC,NDTMPN .... the model name and node number associated with DTMPCC
NATMPC,NATMPN .... the model name and node number associated with ATMPCC
NDRLXC,NDRLXN .... the model name and node number associated with DRLXCC
NARLXC,NARLXN .... the model name and node number associated with ARLXCC
NCSGMC,NCSGMN .... the model name and node number associated with CSGMIN
NCGMAC,NCGMAN .... the model name and node number associated with CSGMAX
NEBALC,NEBALN .... the model name and node number associated with EBALNC
ESUMIS ..... Energy into a submodel
ESUMOS ..... Energy out of submodel

Figure 4–1 shows the general flow for explicit routines for each time step.* The arithmetic nodes are solved iteratively until |ARLXCC| .LE. ARLXCA. On the other hand, Figure 4–2 shows that the implicit routines solve the entire network iteratively for each time step. In addition, the implicit routines also check for |DRLXCC| .LE. DRLXCA before terminating the loop. Both classes of routines, however, perform checks on the maximum temperature change permitted during the entire time step and reduce the time step and repeat the calculations if the checks fail.

Steady state routines do not normally operate as a function of time. They perform iterations until the relaxation criteria ARLXCA and DRLXCA are satisfied (Figure 4–3), whereupon they begin to examine the best criterion for a steady state solution, the relative system energy balance. When the difference between the energy entering the network and the energy leaving it is less than or equal to EBALSA times the energy coming into the system, the iterations stop. If the iteration count had exceeded NLOOPS before the energy balance was achieved, the solution would have terminated at that point. Note that EBALSA is a fractional number, typically 0.01 or 0.02, and represents a relative (unitless) measure of imbalance.

For transient solutions, the relaxation criteria are specified in terms of temperature change per iteration, and the temperature change criteria are specified in terms of temperature change per time step. If the former (DRLXCA and ARLXCA) are not satisfied, then further iterations, up to a maximum of NLOOPT, are performed. If the latter (DTMPCA and ATMPCA) are not satisfied, then the time step is shortened and the calculations are repeated.

* This discussion is limited to the thermal network solutions. For a discussion of the simultaneous fluid network solution, see appendix E.

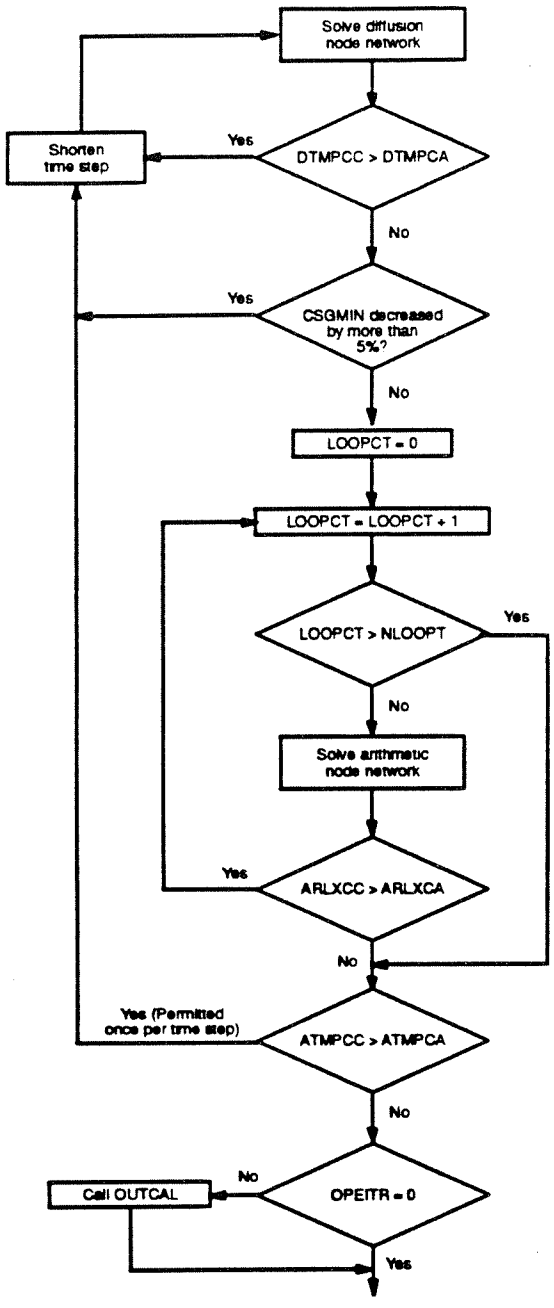


Figure 4–1 Basic Flow in FORWRD and FWDMTS During Each Time Step

The points above will prove useful in understanding the following discussions of the individual, user-specified control constants. Table 4–1 summarizes the constants used by the various network solution routines and shows the default values assigned to each if they are not specified by the user.

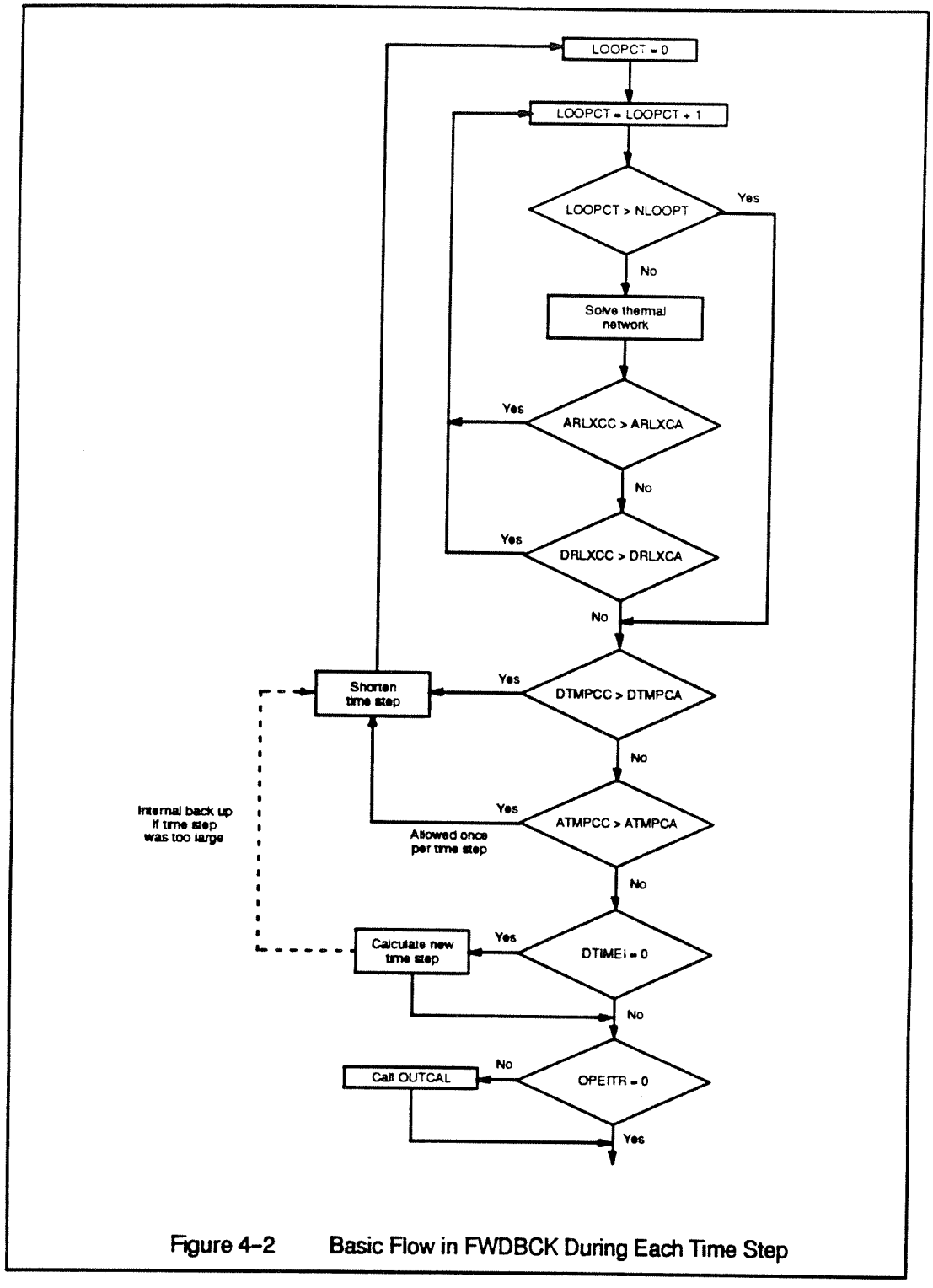


Figure 4–2 Basic Flow in FWDBCK During Each Time Step

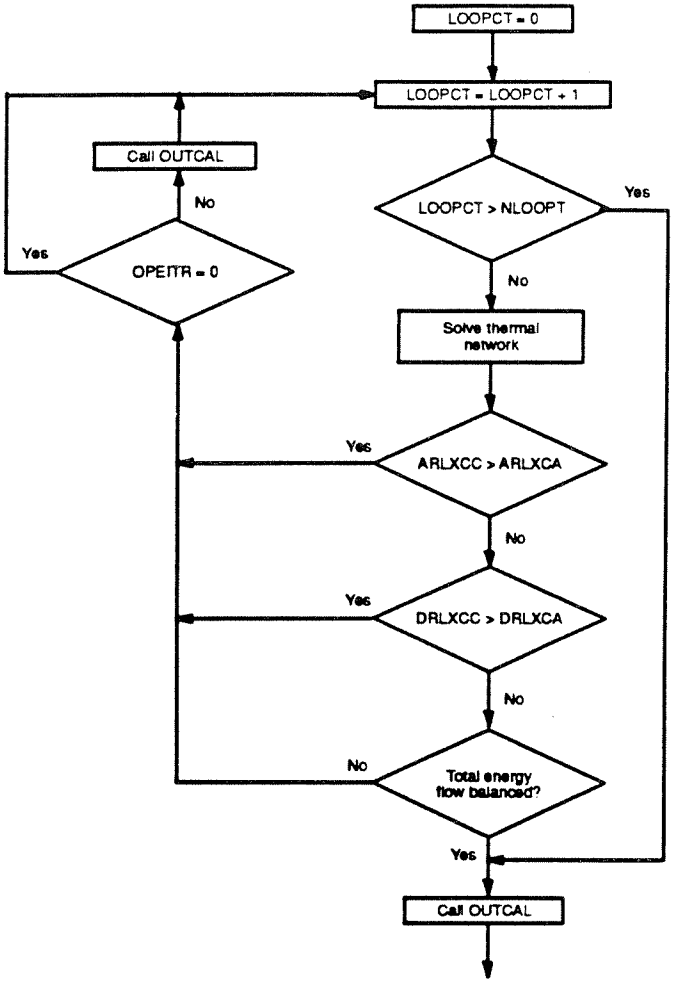


Figure 4–3 Basic Flow in STDSTL and FASTIC

Table 4–1 User-Specified Control Constants Required by Solution Routines

Name	Steady State	Transient	
	STDSTL, FASTIC	FWDBCK	FORWRD
TIMEO	0.0	0.0	0.0
TIMEND	0.0	0.0	0.0
NLOOPS	R,F	NA	NA
DTIMES	0.0	NA	NA
ABSZRO	-459.67 or -273.15*	-459.67or -273.15*	-459.67 or -273.15*
PATMOS	0.0	0.0	0.0
SIGMA	1.0	1.0	1.0
ACCELX	0.0	0.0	0.0
ACCELY	0.0	0.0	0.0
ACCELZ	0.0	0.0	0.0
ARLXCA	0.01	0.01	0.01
NLOOPT	NA	100	100
DRLXCA	0.01	0.01	NA
EBALSA	0.01	NA	NA
EBALNA	0.0	NA	NA
ATMPCA	NA	1.0E30	1.0E30
DTMPCA	NA	1.0E30	1.0E30
CSGFAC	NA	NA	1.0
DTIMEL	NA	0.0	0.0
DTIMEH	NA	1.0E30	1.0E30
DTIMEI	NA	0.0	NA
ITEROT	0	NA	NA
OUTPUT	NA	R	R
ITHOLD	10	NA	NA
EXTLIM	50.0	50.0	50.0
ITERXT	3	3	3
BACKUP	0	0	0
OPEITR	NA	0	0
DTSIZF	NA	0.1	0.1
DTTUBF	NA	0.1	0.1
DTMAXF	NA	1.0E30	1.0E30
DTMINF	NA	0.0	0.0
OUTPTF	NA	F	F
RSSIZF	0.5	NA	NA
RSSIZF	0.5	NA	NA
RSMAXF	1 hour	NA	NA
RERRF	0.01	NA	NA
REBALF	0.01	NA	NA
ITROTF	0	NA	NA
ITHLDF	10	NA	NA
OPITRF	NA	0	0

where:
R ..... required value (if thermal submodels present)
NA ..... does not apply
F ..... required only if fluid submodels present
* ..... default depends on value of UID.

ABSZRO (Value of absolute zero in user units)—This control constant is usually used to define temperature units and is usually set to 0.0, -459.67, or -273.15 although any number may be used. ABSZRO is used internally in all execution routines as an offset factor for all temperatures (both nodes and lumps). The default value depends on the unit system identifier, UID, and is -459.67 if that flag is not used. ABSZRO should not be changed during processor execution.

ARLXCA (Allowable Arithmetic Node Relaxation Temperature Change)—This control constant can be specified for any solution routine in which an arithmetic node is present. If ARLXCA is not specified, the default is 0.01. ARLXCA represents a temperature convergence criterion for the arithmetic nodes during each iterative calculation. Iterations proceed until either all convergence criteria including ARLXCA have been satisfied, or until the maximum number of iterations have been attempted (NLOOPT or NLOOPS). ARLXCA has units of degrees. Although the default is 0.01, its recommended value is dependent upon the magnitude of expected temperatures. The 0.01 value signifies fifth decimal place accuracy for absolute temperatures in the hundreds, but only third to fourth place accuracy for cryogenic temperatures. (See also DRLXCA.)

ATMPCA (Allowable Arithmetic Node Temperature Change)—This control constant may be optionally specified by the user for the implicit or the explicit routines. The default is 1.0E30. ATMPCA represents the maximum allowable arithmetic node temperature change during one time step. It is compared to the calculated temperature change which is stored in control constant ATMPCC. If ATMPCC is greater than ATMPCA, the time step, DTIMEU, is shortened to:

$$DTIMEU = X * DTIMEU*(ATMPCA/ATMPCC)$$

where X = 0.95 for FORWRD and 0.5 for FWDBCK

and the computational procedure is then repeated with the smaller time step. Specification of ATMPCA prevents a rapid temperature change between time steps, with the value to be specified dependent upon the problem. ATMPCA should be used to prevent missing important temperature-dependent variations, and so should be chosen by the user according to the rates of change of properties with temperature. For typical spacecraft thermal problems, an ATMPCA of about 10°F has been found reasonable. *However, it is not always possible to obey the limit imposed by ATMPCA* since arithmetic node temperature changes can be independent of time. *The program will not reset the time step twice in a row because of this criterion to avoid an infinite loop.* Thus, the user may occasionally notice a value of ATMPCC that is higher than ATMPCA.

BACKUP (Backup Switch)—Control constant BACKUP provides the user with the means for utilizing any numerical solution subroutine as a predictor program. All of the numerical solution subroutines set control constant BACKUP to zero just prior to the call on VARIABLES 2. Then, immediately after the return from VARIABLES 2, a check on BACKUP is made. If BACKUP is nonzero, all temperature calculations for the just completed time step are eliminated; the old temperatures (temperatures calculated at the previous time step) are placed in the temperature locations; and control is routed to the start of the computational sequence.

It should be noted that the user must provide the necessary criterion and check in VARIABLES 2 if the iteration is to be repeated. Thus, if the iteration is to be repeated, BACKUP must be nonzero and a criterion that can be met in the next pass must be established to avoid an infinite loop.

Finally, note that only thermal submodels are affected by BACKUP. Fluid submodels cannot be reversed once solved due to memory considerations.

CSGFAC (Time Step Factor)—This control constant may be optionally specified by the user for the explicit transient routine FORWRD. It provides the user with some degree of control over the computer time step. If CSGFAC is not specified, or if it is specified to be less than one, it is internally set at 1.0. For subroutine FORWRD, which is conditionally stable, CSGFAC is a time step divisor; a value of CSGFAC *greater than one* is input to obtain higher accuracy.

DRLXCA (Allowable Diffusion Node Relaxation Temperature Change)—This control constant can be specified for the implicit routine FWDBCK and for the steady state routine. The default value is 0.01. DRLXCA has units of degrees. DRLXCA serves the same purpose for the diffusion nodes as the control constant ARLXCA serves for the arithmetic nodes. Thus, the discussion on ARLXCA hold equally true for DRLXCA. Separate relaxation criteria for diffusion and arithmetic nodes are available because they provide greater computational flexibility and facilitate the coupling of two distinct networks.

While most users use the same value of DRLXCA in both FWDBCK and the steady state routines (STDSTL and FASTIC), this is not always a good practice. The reason while the error in temperatures for steady state routines in on the order of DRLXCA and ARLXCA, this same error applies to each time step taken in FWDBCK. *Thus, the total error in FWDBCK is cumulative using DRLXCA as a criterion.* While the automatic time step predictor (used when DTIMEI=0.0) coupled with the DTMPCA option usually eliminates such problems, the user should consider using a smaller value of DRLXCA in FWDBCK than in STDSTL.

DTIMEH (Maximum Time Step Allowed)—This control constant may be optionally specified for the explicit and the implicit routines. DTIMEH represents the maximum time step allowed during the computational process. If DTIMEH is not specified, it is set to 1.0E+30. When DTIMEI is not input with FWDBCK, then the smaller of DTIMEH, OUTPUT, DTMAXF, and OUTPTF will be the upper limit on the time step used. One of these control constants should be used to make certain that the time step is not allowed to be so large that fast changes in boundary conditions are missed. (See also DTMPCA.)

DTIMEI (Input Time Step for implicit routines)—This control constant is optional for implicit routines and is not used by explicit routines. If DTIMEI is set to 0.0, (the default value) FWDBCK will use the routine FBECHK to choose a time step. *This is the suggested mode of operation; nonzero DTIMEI should be considered only after having tried zero DTIMEI with an appropriate value of DTIMEH.* During execution, FBECHK will vary the time step to maintain the same level of accuracy. It will take large time steps when temperature changes are slow and short time steps when any temperature change is rapid. In almost all cases, this will result in shorter execution time and more accurate results than if DTIMEI is specified. The concurrent use of DTMPCA is recommended. Also, note that DTIMEI is ignored if fluid models are active and if they require a smaller time step.

DTIMEI represents an arbitrary time step, but the governing criteria should be minimum computational time with satisfactory temperature accuracy. Remember that *implicit* means unconditionally *stable* for any time step, not unconditionally *accurate*. In fact, implicitness guarantees nothing about the accuracy of the results, only that they will not diverge if the "real" solution does not diverge. Thus, using an arbitrary time step will result in arbitrary accuracy; nonsensical values may result if DTIMEI is too large. If savings in computational time cannot be met with the same accuracy by using the implicit routines, it is more reasonable to use the explicit routine.

DTIMEL (Minimum Time Step Allowed)—This control constant is optional for the transient solution routines. DTIMEL represents the minimum time step allowed. If the calculated time step is less than DTIMEL, the run terminates with an error message printout. Note DTIMEL is ignored in FWDBCK if the user inputs the time step using DTIMEI. If fluid submodels are active, the fluid control constant DTMINF should be used instead of DTIMEL because that control constant is more tolerant of occasional small time steps.

DTIMES (Steady-state Pseudo Time Step)—This control constant is optional for the steady-state solution routines, although an caution will be produced if TIMEND and TIMEO are not equal because the program then expects usage of the DTIMES option. DTIMES represents an artificial time step which will be taken until TIMEO is incremented to TIMEND. A complete steady-state analysis will be run at each time interval, representing a "transient" analysis based on steady-state assumptions—the system is assumed to react infinitely fast to changes in boundary conditions. DTIMES defaults to zero, meaning that only one steady-state analysis is required at the current time value (TIMEO).

DTMPCA (Allowable Diffusion Node Temperature Change)—This control constant may be optionally specified by the user for the transient solution routines. The default is 1.0E30. DTMPCA represents a maximum allowable diffusion node temperature change between one time step and another. If the maximum diffusion node temperature change, which is stored in DTMPCC, is greater than DTMPCA, the time step is shortened to:

$$\text{DTIMEU} = X * \text{DTIMEU} * (\text{DTMPCA}/\text{DTMPCC})$$

where X = 0.95 for FORWRD and 0.5 for FWDBCK

and the temperatures are reset to their former values. The computational procedure is then repeated with the smaller time step. DTMPCA serves the same purpose for the diffusion nodes as control constant ATMPCA provides for the arithmetic nodes. Unlike ATMPCA, DTMPCA is always obeyed and should be the primary tool for making sure that rapid temperature-dependent variations are not missed. A value of 10°F has been found to be appropriate in most spacecraft thermal analyses.

EBALSA (System Energy Balance Convergence Criterion)—This control constant is used in STDSTL and FASTIC but not in the transient solution routines. The default value of EBALSA is 0.01. Note that the value of EBALSA is expressed as a fraction. This fraction represents an acceptable *relative* energy balance of the system, and is therefore unitless: it is independent of the numerical value of the energy flow through the system. Iterations will continue until the total system energy balance is within the fraction EBALSA of the energy into the system, or until NLOOPS has been reached. Mathematically, the energy balance is achieved when:

ABS(TSUMIS-TSUMOS) .LT. EBALSA * ABS(MAX(TSUMIS,TSUMOS))

where:

TSUMIS ..... Energy into total system (sum of ESUMIS for all submodels)
TSUMOS .... Energy out of total system (sum of ESUMOS for all submodels)

This energy flow check occurs at the master model level only; *checks at the submodel level are neglected due to intermodel conductors.* (The values of ESUMIS and ESUMOS for each submodel are therefore rarely equal.) Furthermore, the system may be considered converged if the total energy flows (TSUMIS and TSUMOS) are unbalanced but are not changing and are not growing towards each other. A caution is produced in such cases, which often arise from one-way conductors, in which case the message can be ignored. However, this message is also encountered when huge conductors are used, and the program has detected negligible progress toward a solution. In the latter case, the user should either reduce the convergence criteria and continue, or consider changes or corrections to the model (such as combining nodes to eliminate the huge conductors, or starting with better initial conditions).

EBALNA (Nodal Energy Balance Convergence Criterion)—This control constant is used in STDSTL and FASTIC but not in the transient solution routines, and is analogous to EBALSA but occurs at the nodal level rather than the master model level. *The default value is 0.0, meaning that nodal level checks are skipped.* Unlike EBALSA, EBALNA is *not* expressed as a fraction, and has units of power. This fraction represents an acceptable *absolute* energy balance of the node. In other words, the value of EBALNA is dependent on the numerical value of the energy flow through the node. Iterations will continue until the largest value of | EBALNC | for all nodes is less than EBALNA of the energy into that node, or until NLOOPS has been reached.

EXTLIM and ITERXT—These two control constants control the behavior of the temperature extrapolation routine. All the solution routines perform temperature extrapolations to speed convergence (this extrapolation is limited to arithmetic nodes in FORWRD). This approach is safer and less model-sensitive than the underdamped acceleration method used in older versions of SINDA.

EXTLIM is the maximum temperature change allowed as a result of an extrapolation. The default value of EXTLIM is 50.0. *Note that this is unit-specific and should be tailored for each model. If a model is oscillating, convergence can sometimes be achieved by reducing this value.* Also, the default value is usually too large if fluid submodels are active.

ITERXT is the number of iterations which must be performed before an extrapolation is performed. The default value of ITERXT is 3. *This is also the minimum value of ITERXT.* If a model is oscillating, convergence might be facilitated by increasing this value. Entering a value greater than NLOOPS or NLOOPT will eliminate extrapolation completely.

Usually the default values of these constants will suffice. The only reason to change them is in case of nonconvergence, in which case EXTLIM should be decreased and/or ITERXT increased. Varying these constants will not force an unstable model to converge and will usually increase the number of iterations required to reach convergence. The choice of which constant to vary and by how much is model dependent.

ITEROT (Iteration Output interval)—This control constant serves as a switch that controls the calling of subroutine OUTCAL for thermal submodels in steady-states. If set nonzero, OUTCAL will be called every ITEROT iteration steps. Setting ITEROT to 1 before NLOOPS is reached (say at NLOOPS=10) is a common method for investigating nonconvergence in troublesome models.

ITHOLD (Number of Iterations to Hold semi-converged submodels)—This control constant prevents thermal submodels from prematurely converging in STDSTL and FASTIC runs by intermittently suspending those submodels that are converging the fastest. The goal is to have all submodels reach final convergence at the same time without wasting computational effort. If a thermal submodel converges *or at least achieves one convergence criterion* (temperature change or energy balance), it is dropped from the computation cycle for ITHOLD (default = 10) iterations and then reactivated until it again satisfies at least one criterion. Final convergence occurs when *all* submodels (thermal and fluid) satisfy *all relevant* criteria. ITHOLD works in concert with ITHLDF, the equivalent control constant for fluid submodels.

Note that the use of ITHOLD can easily increase the total number of iterations, LOOPCT, required to achieve convergence over that required when ITHOLD=0. *This should not be mistaken as an increased cost.* Rather, measurement of the total CPU time would reveal that while LOOPCT increased, the total CPU time *decreased* using ITHOLD since not all submodels were active for all iterations.

NLOOPS and NLOOPT (Number of Iteration Loops)—NLOOPS is the maximum number of iterations for both thermal and fluid submodels in STDSTL and FASTIC. NLOOPT is the maximum number of iterations per time step for the transient routines. The output constant LOOPCT contains the current (or final) iteration count in all routines.

The default for NLOOPT is 100. There is no default for NLOOPS, which *must* be specified for STDSTL and FASTIC. If NLOOPS is not specified, the run terminates with an error message printout. NLOOPT may be specified for the explicit routines since it is used for the arithmetic nodes. For a large steady state problem, it is not unusual to have NLOOPS equal to several hundred to a thousand, whereas for the implicit routines, NLOOPT should be no larger than 100 if DTIMEI is not specified. In fact, in FWDBCK a value of LOOPCT beyond about 10 is an indication of an attempt to take a time step that is too big. For the explicit routines, a value of 100, in combination with an ARLXCA value of 0.01, is usually adequate. In general, a trial and error procedure is required to arrive at suitable values for ARLXCA, DRLXCA and NLOOPT.

When making the initial run of a large steady state problem, NLOOPS should be set to 100 to 200 and ITEROT should be set to 20 to 50 iterations to examine convergence behavior. If the model is obviously converging, it can then be restarted with a large value, say 1000, for NLOOPS.

OPEITR (Output Each Iteration)—This control constant serves as a switch that controls the calling of subroutine OUTCAL for thermal submodels in transients. If set nonzero, OUTCAL will be called at each time step (the term "iteration" is a misnomer), as shown in Figures 4–1, 4–2, and 4–3. OPEITR is mostly used as a model debugging tool.

OUTPUT (Time Interval for Activating OUTPUT CALLS)—This control constant must be specified for all transient solution routines if thermal submodels are present. The input value is left to the judgment of the user. Normally, the output interval is gauged by the length of the run and the expected temperature response characteristics. As a "rule-of-thumb" the output interval lies between CSGMIN and CSGMAX, with OUTPUT being several times larger than CSGMIN. The values of CSGMIN and CSGMAX can be obtained from the output subroutines. When using FWDBCK, the user should remember that OUTPUT or DTIMEH set the upper limit on the time step used (see discussion on control constant DTIMEH), unless fluid submodels are active.

SIGMA (Stefan-Boltzmann Constant)—This control constant provides a convenient means of input for the Stefan-Boltzmann constant, however any number may be used. SIGMA is used internally in all execution routines as a multiplication factor for all radiation conductors. It is never stored in the conductance (G) array. When SIGMA is not specifically input, it has the default value of 1.0. SIGMA should not be changed during processor execution.

TIMEND (Problem Stop Time)—The use of this control constant is self-explanatory. TIMEND must be specified as larger than TIMEO, otherwise an error message is printed and the run terminated. For the explicit routines, if TIMEND is not larger than TIMEO a time step of zero will result and the "TIME STEP TOO SMALL" error message will be printed. FWDBCK will print the message "TRANSIENT TIME NOT SPECIFIED." If a user has some criterion for terminating an analysis, but not the run, he may accomplish this by setting TIMEND=TIMEO when the criterion is met. This will return control to the OPERATIONS DATA block.

If TIMEND>TIMEO in a steady-state run, a nonzero value of the DTIMES constant is expected, and a caution will be otherwise issued.

TIMEO ("Old" Time or Problem Start Time)—This control constant represents the "old" time at the beginning of the current computation interval. It will be updated by the program after each time step is completed, however, *its initial value may be set by the user and will be interpreted as the problem start time.* TIMEO may be set negative, but must be larger than TIMEND. If not specified by the user, its initial value will be 0.0.

4.7 Fluid System Modeling

This section is intended both as an introduction to FLUINT (*fluid integrator*) and as a source of detailed explanations of fluid modeling methods. Note that FLUINT and SINDA are actually two halves of one program (which consists of one preprocessor and one processor library). Any reference in this section to *FLUINT* means *the fluid network capabilities in SINDA/FLUINT*, while *SINDA* means *the thermal network capabilities*.

The reader should refer to Section 3.11 for details regarding input formats. Sections 4.7.1 through 4.7.4 introduce the reader to fluid modeling concepts, and the approach that the analyst should take when thinking of a fluid system in terms of a FLUINT model. Sections 4.7.5 through 4.7.7 describe the features and options that facilitate program use. Sections 4.7.8 through 4.7.15 describe more advanced options available for specialized modeling requirements. Section 4.7.16 describes ways to streamline models for efficient program usage. Section 4.7.17 lists the available output routines. Section 4.7.18 describes the FLUINT accuracy control variables and the execution routines that are called from the user logic blocks, and describes the steady-state convergence process and the transient time step selection criteria. Section 4.7.19 describes how to resolve common modeling errors.

4.7.1 FLUINT Scope and Goals

FLUINT provides a general analysis framework for internal one-dimensional fluid systems. It can be used alone or in combination with the thermal analysis capabilities of SINDA. FLUINT is to thermal/hydraulic analyses what SINDA is to conduction/radiation analyses. The specific capabilities are described in the following paragraphs.

Arbitrary Fluid Systems. FLUINT is not restricted to any specific geometries or configurations. The program is based on network methods, allowing the user to describe almost any schematic representation.

Fluid Independence. FLUINT may be used to analyze any working fluid with adequately defined properties. Twenty refrigerants are immediately available, and the user may describe the properties of additional gases, liquids, and two-phase fluids with reusable FPROP DATA blocks.

Variable Resolution and Assumption Levels. FLUINT may be used for a wide variety of analyses, ranging from first-order performance approximations to detailed transient response simulations. Spatial resolution may be varied by adding or subtracting network elements, and temporal resolution may be varied by choosing between time-dependent and time-independent types of elements. Furthermore, the performance of fluid components such as pumps can be as idealized or as realistic as appropriate to the problem and to the level of known detail.

Steady-State and Transient. FLUINT can be used for either steady-state analyses or for transient analyses. As noted above, the fidelity of the transient analysis can be varied from a time-independent thermal response (i.e., quasi steady-state hydrodynamics) to a fully time-dependent thermal/hydraulic response (i.e., acoustic waves and water hammer), depending on the *type* of network element selected.

Single and Two-Phase Flow. The code handles both single-phase and two-phase flow, along with transitions between states. This generality exists not only in the numeric solution, but also in the correlations used for each device or phenomena. In fact, the capability exists to handle capillary devices, where surface tension forces dominate when two phases are present. Within the realm of two–phase flow, the phases are by default in equilibrium with each other and travel at the same velocity (i.e., homogeneous), but either assumption may be avoided if necessary.

Component Models. FLUINT includes general device models for a wide variety of common fluid system components. The user can tailor these models for specific devices. The user can even model new or unique devices by accessing program variables with his own, simultaneously executed Fortran logic. The basic building block style network elements were specifically designed for such usage.

Similarity to and Compatibility with SINDA. The basic methods in FLUINT contain many analogies to the lumped-parameter style of SINDA, which should help the SINDA user learn FLUINT easily. FLUINT may be thought of as a fluids co-processor to SINDA. Analyses may be all-fluid, all-thermal, or both fluid and thermal simultaneously. FLUINT, along with SINDA, dynamically sizes the processor for minimum overhead, and allows user control and user logic.

FLUINT Limitations—The user should be also aware of current program limitations. These limitations can be divided into two classes. First, there are some limitations that may be circumvented by extensive user logic if the user is sufficiently experienced, but which are not yet available as prepackaged options. Second, there are certain program capabilities that cannot be appended within user logic, but may eventually be added to FLUINT as growth options. The subsequent paragraphs will briefly mention alternatives for advanced users if any alternatives exist.

Internal Fluid Systems. The program is intended for 1D piping networks and can be used on certain classes of 2D flow problems, but not for external flow or 3D internal flow.

Working Fluid Choices. Although the code can analyze multiple fluid systems each containing different working fluids, no mixtures are permitted—only one fluid substance may exist in any one system. Furthermore, the internal correlations and methods are not applicable to superfluid helium or liquid metals, although the correlations can be altered and the axial conduction terms accounted for by advanced users. Any other one-substance or one-phase fluid can be analyzed, providing the user develops a reusable description of the fluid if it is not one of 20 prestored working fluids. The effect of adjacent (but non–dissolvable) fluids such as noncondensible gases can be accounted for using the NCGBUB routine if the gases are stationary. Also, the MIXGAS routine can be of use simulating the mixing of separate perfect gases contained within different fluid submodels. Other multiple–species methods include the use of multiple fluid submodels combined with a partial pressure assumption.

Low Velocities. The code is intended for low-speed, viscous flows in ducts. Mach numbers should not exceed about 0.4 in duct segments. Certain compressible flow phenomena such as traveling shocks are not included, although certain effects like valve choking and acoustic wave propagation can be simulated. While density need not be constant, the program does not make the distinction between static and dynamic temperatures and pressures.

Liquid Compressibility. Liquids are assumed incompressible, meaning that their densities are functions of temperature only. Therefore, the program is not strictly intended for water hammer or column separation analyses. However, the compressibility of liquid *can* easily be

included in the system response by making the control volume walls as compliant as the volumetric modulus of the liquid within the volume. Liquid compressibilities and speeds of sound are available for such calculations, along with routines (COMPLQ, Section 6.9.6.7) that facilitate such an approach. This approach is not intended for pipe-burst or external loading studies because pressure profiles may be nonconservative due to the implicit control volume approach used within the code. Nonetheless, excellent comparisons have been made with codes that are specifically intended for water hammer analyses as long as the time step is kept sufficiently small (refer to the WAVLIM routine in Section 6.9.6.7).

4.7.2 The Fluid Submodel Concept

FLUINT introduces a new type of submodel: the *fluid submodel*. All traditional SINDA models composed of nodes and conductors are called *thermal submodels*. Master models may be composed of fluid submodel(s), thermal submodel(s), or both. No single submodel may have attributes of both thermal and fluid submodels; a coupled thermal-fluid analysis must therefore have at least one submodel of each type. As with thermal submodels, fluid submodels are semi-independent entities that have their own inputs, logic, and control. Provisions have been made, however, for allowing heat energy to pass between fluid and thermal submodels, as will be described later.

There is one important difference between the two types of submodels: while thermal models may be divided and combined arbitrarily into submodels and still produce the same results, fluid submodels are distinct and indivisible. While thermal submodels may be interconnected by common conductors, fluid submodels cannot exchange fluid with one another. The rule of thumb for determining what should be modeled as a fluid submodel is therefore "one working fluid, one submodel." However, if two fluid systems use the same working fluid description but do not share or exchange fluid, they should still be input as separate submodels for efficiency. This is not considered a restriction in usage because it enforces a logical breakdown for complex, multi-fluid systems.

One last distinction must be made between the two types of submodels. Fluid submodels are specialized to thermal/hydraulic analyses and have no capability for handling conduction or radiation. Thermal submodels, in turn, are specialized for conduction and radiation, and do not adequately handle fluid analysis tasks. Therefore, the scope of the fluid model is the physics *within* the fluid container walls, while the scope of the thermal submodel is the physics *outside of and perhaps including* the fluid container walls. This specialization is a key ingredient in the efficiency and generality of SINDA/FLUINT.

FLUINT automatically calculates fluid properties. Therefore, the code must know which system of units is being used, and all submodels (both fluid and thermal) must use these units. Conversion factors may be used to translate portions of the model into the master unit set.

4.7.3 Networks and Elements

The longevity and widespread use of the SINDA–family of thermal network analyzers is testimony to their flexibility. Their approach is called *lumped parameter*, whereby a thermal problem is treated by breaking the problem into two halves: energy transport (conductors) and

energy storage (nodes). (Arranged in orderly meshes, the lumped parameter equations represent the *finite difference* approximation to the conduction equations.) Nodes and conductors are the elements with which mathematical models of real thermal systems are constructed. Nodes communicate with each other via conductors in carefully constructed *networks*. This allows the real system to be approximated with any degree of resolution desired. However, from the program's viewpoint, the organization of networks cannot be predicted beforehand, so the program must be organized to handle "arbitrarily" constructed networks of the basic elements.

FLUINT works the same way. However, in addition to energy there are two more variables in a lumped parameter approach for fluid networks: mass and momentum. This considerably complicates the analysis, and subsequently increases the amount of input, the number of variables, and the computational cost per element. However the basic approach is the same: FLUINT treats a fluid problem by breaking it into two halves: energy and mass transport (*paths*), and energy and mass storage (*lumps*). Fluid networks are, therefore, arbitrarily constructed networks of paths and lumps sharing a common working fluid. Energy exchange between thermal and fluid submodels is handled by *ties* between nodes and lumps.

Lumps are the fluid analogs of nodes. While nodes are characterized by temperatures and perhaps thermal capacitances, lumps are characterized by temperatures, pressures, qualities and perhaps volumes. Lumps are assumed to always be perfectly mixed and to exist in thermodynamic equilibrium; *a lump has only one characteristic thermodynamic state.* (To deviate from perfect mixing within two-phase control volumes, two lumps may be used in conjunction with the NONEQ0 and NONEQ1 simulation routines.)

Paths are the fluid analogs of conductors. However, paths can be very complicated, and this analogy is not as close as the one between lumps and nodes. Lumps communicate both mass and energy with each other via the flowrate in paths, while nodes communicate energy with each other via the energy transport in conductors. Path flows are assumed one-dimensional, and phase velocities are assumed equal (homogeneous or zero slip); *a path has only one characteristic flowrate.* (To model slip flow within two-phase ducts, two parallel paths may be input. These pairs are called *twins*.)

To further aid in the distinction between lumps and paths, it should be noted that no specific flow rate or geometric shape can be attributed to a lump, and that no specific thermodynamic state (or mass or heat rate) can be attributed to a path. This is the same as saying that a node does not have a conductance, and that a conductor does not have a temperature. This distinction is important in fluid analysis because of its impact on heat transfer calculations, as described in a later section.

Ties are very much like conductors, in that they allow only heat energy to pass from higher temperature sources to lower temperature sinks according to a conductance based on the desired heat transfer phenomena. *One side of a tie is always a thermal node, and the other is always a fluid lump.* Without ties, fluid and thermal models would be independent of each other.

The user should note that FLUINT always calculates its own maximum allowable time step, and that steady-state solutions may achieved either by taking long artificial time steps until the network is relaxed (STDSTL), or by iteratively relaxing the network (FASTIC). The size of time-dependent elements (tanks and tubes) will determine the size of the real or artificial time step that can be taken—the larger the element, the larger the time step. See Section 4.7.18 for a discussion of time step control.

The following subsections describe the fluid elements in more detail. As a summary, Table 4–2 contains an abbreviated description along with the analogies with thermal elements. Figure 4–4 details the hierarchy of FLUINT elements.

Table 4–2 Summary of FLUINT Elements

ELEMENT	TYPE	ANALOG	DESCRIPTION
Lump	Tank	Diffusion Node	Control volume, time-dependent
	Junction	Arithmetic Node	Zero volume lump, time-independent
	Plenum	Boundary Node	Infinite volume lump, time-independent
Path	Tube	no analog	Line segment with significant inertia, time-dependent
	Connector	Linear Conductor	Time-independent, multi-purpose path (insignificant inertia)
Tie	HTU, HTUS	Linear Conductor*	User input conductance
	HTN, HTNC, HTNS	Linear Conductor*	Conductance calculated according to convection correlations

* perhaps more "equivalent" than "analogous"

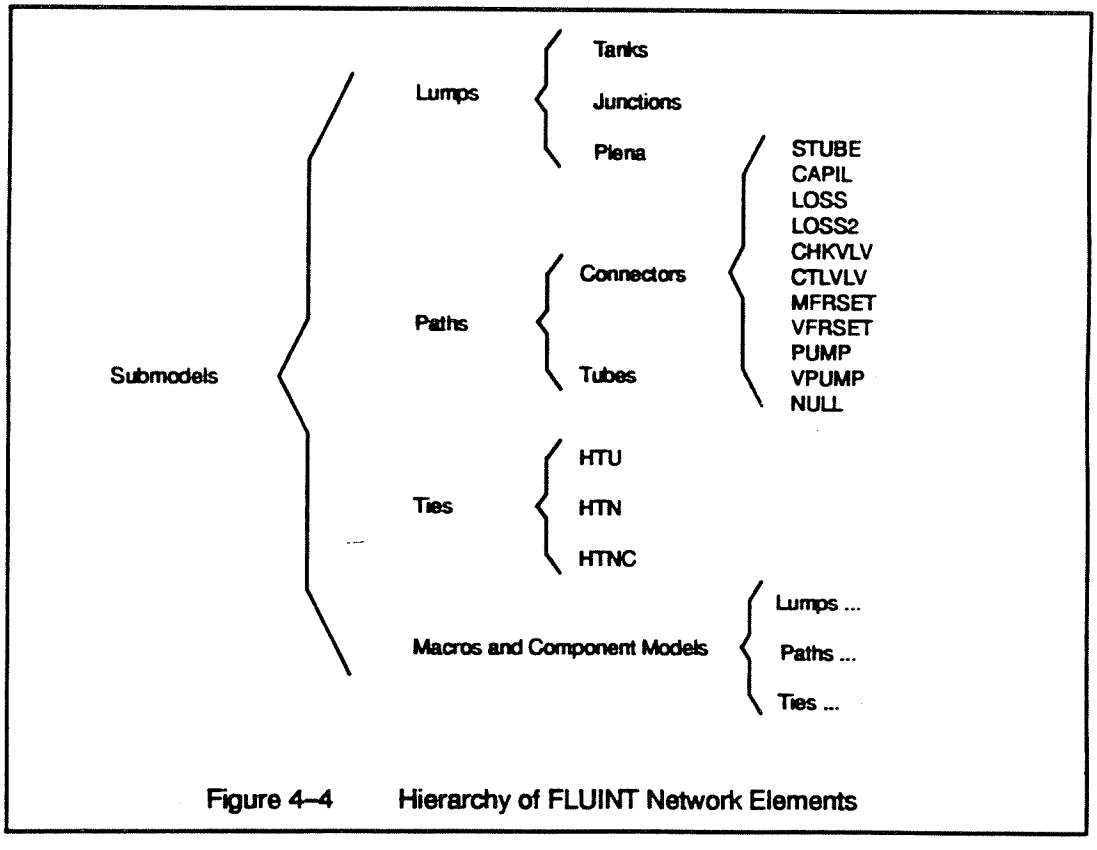


Figure 4–4 Hierarchy of FLUINT Network Elements

4.7.3.1 Lumps

All lumps have a single specific thermodynamic state. This state must be initialized by the user in FLOW DATA. The user can also access to the state in his logic blocks, just as he has access to the temperatures and capacitances of nodes. While five of the properties describing the lump state are available for the user's convenience, there are only three properties needed to determine the lump state initially in FLOW DATA. These input properties are:

TL ..... Lump temperature
XL ..... Quality (vapor mass fraction: 0.0 for 100% liquid, 1.0 for 100% vapor or gas)
PL ..... Lump pressure

While only two properties (TL, PL) are required to completely specify a single-phase state, the pressures and temperatures are not independent for two-phase states. Therefore, a third property, quality (XL), is also needed. Since there is a potential conflict between these three properties for single-phase states (where only two are required), *by default FLUINT assumes that TL and XL are always correct, but that PL may be a first guess and hence will be overridden if it conflicts with TL and XL.* Table 4–3 summarizes the manner in which states are specified. The reason for this method is to allow users to specify all possible states without having to keep track of saturation pressures for two-phase states. Thus, the user can force FLUINT to initialize the lump in a saturated state by providing an unrealistically high PL (say 1.E30) if XL isn't 0.0, or an unrealistically low PL (say 0.0) if XL isn't 1.0. Changes to the state can be made later in OPERATIONS data using the CHGLMP routine before calls to any solution routines.

Alternatively, the user may replace this hierarchy by another using "PL!" instead of "PL" within FLOW DATA blocks. *(PL! is illegal within logic blocks.)* By placing the exclamation mark after the "PL," the user may signal that the input pressure value is to be given priority over the input temperate value, which may be overridden if it conflicts with PL and XL. Thus,

Table 4–3 How to Specify a Lump State (without "PL!")

Desired State	Input Quality	Input Pressure
Subcooled Liquid	XL = 0.0	PL > P_{sat}(TL) *
Saturated Liquid	XL = 0.0	PL ≤ P_{sat}(TL) **
Two–phase	0.0 < XL < 1.0	anything **
Saturated Vapor	XL = 1.0	PL ≥ P_{sat}(TL) **
Superheated Vapor	XL = 1.0	PL < P_{sat}(TL)

* P_{sat}(TL) is the saturation pressure corresponding to TL.
** PL will be replaced with P_{sat}(TL) in these cases.

the user can force the use of a certain initial pressures, to which the program is often more sensitive than initial temperatures. Arbitrarily high or low TL values can then be used to specify saturated states in a manner analogous to Table 4–3.

Note that the fluid submodel initial conditions (both thermodynamic state and flow rates) need not represent real conditions—they may simply be guesses at a true set of initial conditions that are to be passed to the steady-state solution routine. (The solution routine FASTIC is specifically intended to handle such rough guesses.) However, users will generally find it is worth their time to specify reasonable initial conditions—the program may not be able to recover from highly inconsistent or extreme initial conditions.

As mentioned before, there are five variables describing the thermodynamic state that are available within user logic blocks. These are:

TL ..... Lump temperature (user-defined absolute zero)
PL ..... Lump pressure (user-defined absolute zero, *double precision*)
XL ..... Lump quality (ratio of vapor mass to total mass; dryness fraction)
DL ..... Lump density
HL ..... Lump enthalpy (intensive)

These properties should not be altered within user logic blocks without the use of the CHGLMP routine. They are available to the user for control or as input to calculations. There are many such variables in FLUINT. Altering their value may produce unexpected results. *Note that if PL and PATMOS (the pressure analog of ABSZRO) are used in user logic blocks, they should be treated as double precision variables.*

Optionally, lumps may be assigned an elevation or coordinate location (CX, CY, CZ) in cartesian three-space. These coordinates, combined with global acceleration vector components ACCELX, ACCELY, and ACCELZ, are used to calculate body forces resulting from gravity, launches, or orbital maneuvering. These features are described in a later section.

Lumps have additional descriptive variables, depending on type, as discussed below. Recall that there are three basic types of nodes: diffusion (finite capacitance), arithmetic (zero capacitance), and boundary (infinite capacitance). There are three analogous types of fluid lumps: *tanks* (finite volume), *junctions* (zero volume), and *plena* (infinite volume). The differences between these types of lumps are described below.

Tanks—Tanks are fluid lumps with a definite, nonzero volume. Like all lumps, they have specific thermodynamic properties. Unlike other lumps, mass and energy within a tank change gradually with time; *tanks resist most sudden changes*. Therefore, the relative size of a tank might affect the size of the allowable time step—bigger tanks might mean larger time steps during a transient solution. Tank volumes can expand or shrink at prescribed rates (independent of tank pressure), or may expand or shrink as a function of pressure, or both effects can be superimposed. Like the Q-source term in thermal networks, tanks have an analogous source/sink term. The additional descriptors for tanks are:

VOL .... Tank volume
VDOT ... Rate of change of VOL with time (positive for growth)
COMP ... Tank wall compliance (volumetric spring rate with pressure)
QDOT ... Heat input (source term)

QDOT and VDOT may be altered within user logic blocks, but VOL should not be altered except with the CHGVOL routine. QDOT values are automatically calculated for tanks with heat transfer ties to thermal models, but otherwise QDOT can be modified in logic blocks. Tanks may not have negative volumes, so care should be taken in the selection of VDOT. The tank wall compliance may be used to model gas inclusions, container flexibility, and liquid compressibility, as described in later sections.

Tanks can be used in one of two basic ways. First, as their name implies, they can be used to model a vessel, reservoir, or accumulator directly. Second and most important, tanks are control volumes—they can be used to simulate the energy or mass storage of pipe segments, heat exchanger segments, or any portion of the fluid system with significant volume. Note that if all tanks in a submodel are or can become hard-filled (subcooled) and are stiff (COMP=0.0), *at least one plenum must be present.* The pressure in such tanks will change instantly to respond to flowrate, volume, or even heat rate (density) changes since liquid phases are assumed incompressible.

Such hard-filled, noncompliant tanks (*hard tanks*) react much differently than *soft tanks* (i.e., tanks with any vapor). Since the flowrates into hard tanks must always nearly balance, these tanks react faster than soft tanks to changes and their hydrodynamic response can be instantaneous. Because of special treatment, hard-filled tanks can take much larger time steps and consume much less computer time per step than soft tanks. For this reason, soft tanks should be used more sparingly, and junctions should be considered as alternatives. However, pressures may change too quickly or extremely in a hard tank since there is no place to store or release any excess fluid. Tank wall compliance factors (COMP) can be used to reduce this problem by removing the assumptions of inflexible walls and incompressible liquid. Refer to the routine COMPLQ in Section 6.9.6.7.

Tanks may also be used in pairs to avoid the perfect mixing assumption within two-phase control volumes. Refer to the simulation routines NONEQ0 and NONEQ1 in Section 6.9.6.5.

Junctions—Junctions are to tanks what arithmetic nodes are to diffusion nodes—they are "zero volume tanks." In all other respects, a junction is like a tank. It has a specific thermodynamic state and an energy source/sink term, QDOT, which may be altered within user logic blocks unless the junction is tied. However, because it has zero volume, a junction cannot grow or shrink or store mass and energy—junctions are time-independent. All mass and energy entering a junction equals the mass and energy leaving at all times: *junctions react instantly to changes*. They therefore do not directly affect the allowable time step during transients.

Junctions may be used in three basic ways. First, they may be placed at any point in the fluid system where temperatures or pressures are needed, or where heat must enter or leave the system. Second, they may be used as tees, end caps (dead ends), or common points to branching lines. Third, they may be used as control volumes that react instantaneously. In short, they may be used to join paths together wherever the volume is negligible or has been attributed to a nearby tank.

There are three cautions to be used regarding junctions. First, a fluid submodel cannot be built entirely of junctions for the same reasons that a thermal submodel cannot be built entirely of arithmetic nodes.

Second, fatal errors will result if a loop (closed path) exists in the network that is composed solely of junctions and QDOT values in that loop are not zero or do not exactly balance (except in FASTIC). The user should therefore avoid such *junction loops*, examples of which are shown in

Figure 4–5. The reason for this restriction is actually the same reason why thermal models cannot be built entirely of arithmetic nodes: within each recirculating loop, there is nowhere for energy imbalances to be absorbed, causing the temperatures to spiral to positive or negative infinity. Note that junction loops depend on the actual flowrate direction, not the defined flowrate direction—what may not have been a junction loop at the start of the run could become one. Remember, the user need not worry about this restriction if all of the junctions are adiabatic.

Third, tied or diabatic (nonzero QDOT) junctions are sensitive to flowrate changes. To conserve energy between thermal and fluid submodels in transient routines, the QDOT of all tied lumps is held constant over the time interval, as is the corresponding Q term on the thermal nodes. If the flowrate changes by more than a few percent during that interval, property routine error messages may result. While these messages are inconsequential if infrequent or not persistent, they may otherwise signal modeling problems.

Junctions can also be used in a fashion similar to SINDA heater nodes via the HTRLMP routine. Using HTRLMP, the enthalpy of a junction can be fixed (which usually means either the temperature or quality is nearly constant if the pressure does not change much), and the QDOT term is updated to contain the heat rate required to hold that enthalpy fixed. Such "heater junctions" are very useful for modeling oversized heat exchangers, and for several modeling tricks where the actual heat transfer processes are not considered part of the problem. RELLMP may be used to "release" the enthalpy, returning the junction to a normal mode of operation. (HTRLMP may also be used on tanks within FASTIC.)

Plena—Plena are the "boundary nodes" of the fluid network. A plenum has an infinite volume. As such, it represents an inexhaustible source or sink of fluid; the thermodynamic state does not change. Plena do not affect time steps or solution speeds. They have no energy source/sink term. Their descriptors simply consist of the five thermodynamic properties. Plena may be used as boundary conditions for open systems, or as a simplified model of a very large tank whose response is not part of the problem at hand. Note that tanks and junctions may be temporarily made to act like plena using the HLDLMP and RELLMP routines.

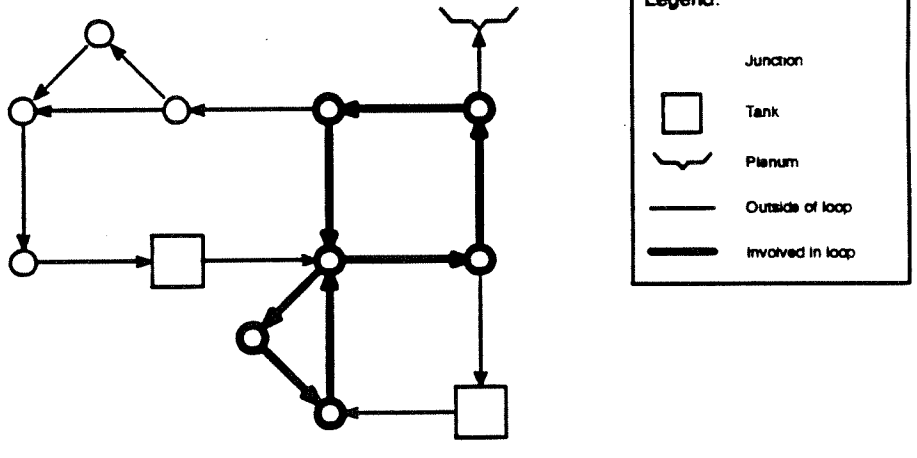


Figure 4–5 Examples of Junction Loops

4.7.3.2 Paths

Like conductors, paths come in two basic types. Paths are either time-dependent (*tubes*) or time-independent (*connectors*). Tubes exhibit inertia, whereas connectors have none and hence react instantaneously to changes. Connectors are somewhat analogous to thermal conductors, but tubes have no thermal analog. Thus, while the thermal-fluid analogy has been useful so far, it will be dropped at this point to minimize confusion.

All paths have at least one descriptive parameter in common: a mass flow rate (FR). This descriptor must be given an initial value, and is available to users in their logic blocks. As with thermodynamic properties in lumps, FR should not be directly altered within user logic blocks once the solution process is under way. It is available to the user for initialization, or as input for calculations such as heat transfer or pressure drop. The positive direction for FR is defined by specifying an upstream and downstream lump. This only determines the sign convention—flow rates may be zero or negative (e.g., reversed). For this reason, the endpoint lumps are sometimes referred to as the *defined upstream* and the *defined downstream* lumps, referring to definition of positive flowrate.

Optionally, paths may be assigned duplication factors, which is a general device for exploiting symmetry to reduce the size of the model. For example, an upstream duplication factor (DUPI) of 10.0 means that 10.0 such paths appear from the perspective of the defined upstream lump even though only one path exists. Discussion of these factors is deferred to a later section.

Paths may also utilize phase-specific suction options, whereby the path extracts only liquid or only vapor from a two-phase lump. This option, exercised by the STAT flag in FLOW DATA and the CHGSUC routine in logic blocks, is described below and in a subsequent section.

To model slip (nonhomogeneous) flow, paths may be twinned, as described in Section 4.7.14. Twins are nearly identical to each other; they share the same endpoint lumps, same type, etc. However, one of the pair is intended to carry liquid when it is available, and the other is intended to carry vapor. These are called the *primary twin* and *secondary twin*, respectively. For tubes and for STUBE connectors (defined below), twinning a path invokes special logic which is intended to be used in conjunction with the flag IPDC=6. This simulation logic and its implications will be described below. For other connector devices, paths may be twinned, but additional simulation logic is not normally invoked. Rather, two nearly identical paths are created, the primary being initialized with liquid–only suction, and the secondary being initialized with vapor–only suction.

Tubes—Tubes are models of fluid lines with nonnegligible inertia. This means that *the flowrate inside a tube cannot change instantaneously*—it takes time for the forces acting on the tube (such as pressures at the ends) to accelerate or decelerate the fluid inside the tube. Thus, tubes are time-dependent and the size of the tube (measured by the fluid density times the length to diameter ratio, $\rho L/D$) might affect the allowable time step during a transient. The density factor means that liquid lines have more inertia than vapor lines. Other approximations are available for small $\rho L/D$ ducts, as discussed below.

Recall that tubes, in spite of being defined by lengths and diameters, do not contain any mass nor can heat be transferred through the walls directly. Any mass associated with a tube should be attributed to the lumps on either end of it, and any energy must similarly be added or subtracted through those lumps. This does not imply that tubes do not have inertia, or that the heat transfer calculations neglect tube shape (see Section 4.7.5 and Appendix B).

There are many descriptive inputs available for tubes, and most may be altered within user logic blocks. The first-time user should not be concerned with the number or apparent complexity of these parameters. Most have convenient defaults, or are calculated automatically. Thus, the rule of thumb for tubes is: *If you don't understand it, ignore it. It will probably go away.* However, experienced users will find that access to so many parameters will stretch the intended usage of FLUINT to many analysis needs.

Table 4–4 summarizes the tube descriptors. They are described in more detail in the paragraphs that follow.

Table 4–4 Tube (and STUBE Connector) Descriptive Parameters

Param.	Req'd* Input?	Avail.**? (FLOGIC)	Can Alter?	Automatic Calculation?	Description
FR	Yes	Yes	No	Yes (solved for)	Mass flow rate
STAT	No	No	Yes***	If twinned	Liquid/vapor suction status
TLEN	Yes	Yes	Yes	No	Tube length
DH	Yes	Yes	Yes	No	Hydraulic diameter
AF	No	Yes	Yes	No	Flow area
AFTH	No	Yes	Yes	No	Throat flow area
IPDC	No	Yes	Yes	No	Friction pressure drop correlation number
UPF	No	Yes	Yes	No	Upstream fraction of properties to use in friction pressure drop calculations
WRF	No	Yes	Yes	No	Wall roughness fraction
FK	No	Yes	Yes	No	Additional K-factor losses
HC	No	Yes	Yes	No	Head coefficient
AC	No	Yes	Yes	If in macro	Recoverable loss coefficient
FC	No	Yes	Yes	If nonzero IPDC	Irrecoverable loss coefficient
FPOW	No	Yes	Yes	If nonzero IPDC	Flowrate exponent in FC term
FD	No	Yes	Yes	If twinned, IPDC=6	Interface drag coefficient
FG	No	Yes	Yes	If in macro, twinned	Phase generation coefficient
AM	No	Yes	Yes	If twinned	Added mass coefficient

* Default can be defined earlier (see PA DEF subblocks)

** In general, refer only to primary twin in logic; secondary twin will assume that value if changed

*** Use CHGSUC routine

The governing differential equation for tubes is:

$$\frac{dFR_k}{dt} = \frac{AF_k}{TLEN_k}\left(PL_{up} - PL_{down} + HC_k + FC_k \cdot FR_k \cdot |FR|_k^{FPOW_k} + AC_k \cdot FR_k^2 - \frac{FK_k \cdot FR_k \cdot |FR|_k}{2 \cdot \rho_{avg} \cdot AF_k^2}\right)$$

This equation is simply Newton's second law relating force and acceleration (F=m•a). The subscript k refers to the current (k^{th}) tube. This equation is presented as an aid in the understanding of certain tube options. It is not necessary to understand this equation to use tubes effectively.

Twinned tubes: Slip Flow Simulation—Tubes may optionally be twinned, or input in matched pairs, to simulate slip flow. The primary twin carries liquid when liquid is present upstream, and the secondary twin (i.e., the one named after the "TWIN=" keyword) carries vapor. In the limits of single phase flow, the secondary tube is ignored and the primary tube carries all the flow, whether liquid or vapor. This option is intended to be used in conjunction with the flow regime mapping option described below, which is invoked by specifying IPDC=6. Refer to Section 4.7.14 for more in–depth discussion of flow regime mapping and slip flow simulation.

The governing differential equation for each member of a pair of twinned tubes is:

$$\frac{dFR_k}{dt} = \frac{AF_k}{TLEN_k}\left[\alpha_k (PL_{up} - PL_{down} + HC_k) + \frac{1}{\alpha_k}\left(FC_k \cdot FR_k \cdot |FR|_k^{FPOW_k} + AC_k \cdot FR_k^2 - \frac{FK_k \cdot FR_k \cdot |FR|_k}{2 \cdot \rho_k \cdot AF_k^2}\right)\right]$$

$$\frac{1}{\alpha_k} FD_k \cdot FR_k - \frac{1}{\alpha_t} FD_t \cdot FR_t + \varepsilon_k\left(\frac{1}{\alpha_k} FG_k \cdot FR_k + \frac{1}{\alpha_t} FG_t \cdot FR_t\right)$$

$$\alpha_t \cdot AM_t \cdot \frac{dFR_t}{dt} - \alpha_k \cdot AM_k \cdot \frac{dFR_k}{dt}$$

Again, this equation is derived from Newton's F=m•a. The suffix t refers to the twin of the current (k^{th}) tube; the t^{th} tube follows the same equation except that the k and t suffixes are reversed. Note that AF_k=AF_t and $TLEN_k$=$TLEN_t$, but that other parameters in the above equation are not shared.

The areal fraction of the k^{th} tube is α_k, where this fraction is defined as the ratio of the phasic cross sectional area to total cross sectional area. If k is the secondary (vapor) tube, α_k is equal to the void fraction although the above definition is intended to cover the parallel liquid volume fraction as well. Also, just as $\alpha_v = 1 - \alpha_l$, then by definition $\alpha_k = 1 - \alpha_t$. The above equation was derived from the simple homogeneous tube equation by multiplying all all pressure (driving) terms by α, and by dividing all velocity (response) terms by α.

The term ε_k, whose value is either +1 or –1, is defined below in the paragraph describing FG.

Tube (and STUBE connector) Parameters—The following paragraphs describe the parameters that comprise the tube governing equations. STUBE connectors, which will be described later, use a slightly modified version of this equation with the same parameters. Therefore, the following discussion is applicable to both tubes and STUBE connectors, except as noted.

FR: Mass flow rate—This parameter has been previously described. The initial value of FR should be chosen carefully (unless FASTIC is called) since it will affect solution speed. Zero flow rates should only be input where they are expected. Small flowrates adversely affect time steps because the flowrate changes per time step are often restricted to a percentage of the total current flowrate. Once initialized (in FLOW DATA or in OPERATIONS DATA), the FR should not be changed directly. (Indirect changes may be made via other parameters, such as the SMFR factor for MFRSET connectors, which are described in later sections.)

STAT: Liquid / vapor suction status—The assumption of homogeneous phase distribution in two-phase lumps may be avoided using this option. STAT can only be altered during execution using the CHGSUC routine. The use of the STAT option is not restricted to tubes. With a few exceptions, it can be applied to any path.

The default value is STAT=NORM. Under this condition, a tube will extract both liquid and vapor from a two-phase upstream lump. STAT can be set such that the tube will extract only liquid or only vapor from such a lump. This is useful in the simulation of stratified tanks, liquid/vapor separators, and capillary devices. The following list details the results caused by different STAT values:

NORM .. Homogeneous (suction of both phases - default)
LS ..... Liquid-only suction from defined upstream end
VS ..... Vapor-only suction from defined upstream end
RLS ..... Liquid-only suction from defined downstream end
RVS ..... Vapor-only suction from defined downstream end
DLS ..... Liquid-only suction from either end
DVS ..... Vapor-only suction from either end

An important restriction applies to STAT operation: liquid-only or vapor-only suction only applies when the upstream lump is in a two-phase state. Otherwise, the tube extracts whatever phase is available even if that phase is not the one specified in the STAT option. Phase-specific suction from junctions (or tanks in FASTIC) can produce counterintuitive results, as described in later sections.

The STAT option is manipulated internally by twinned paths, and should not be used in combination with them unless they have been forced to stay homogeneous using the GOHOMO routine.

TLEN: Tube length—The tube length is a required input that is available in the user logic blocks and may be altered if necessary. It should be noted that the "size" of the tube is directly proportional to TLEN. This is important information to keep in mind when developing a fluid model, since tube size can directly affect the allowable transient time step. TLEN is also used to determine frictional pressure drops if IPDC is nonzero, and may also be used in other calculations such as heat transfer.

DH: Hydraulic diameter—The hydraulic diameter is a required input that is available in the user logic blocks and may be altered if necessary. The hydraulic diameter is defined as four times the flow area divided by the wetted perimeter. This reduces to the diameter for a circular cross section, and to the length of one side for a square cross section. DH is used to determine frictional pressure drops if IPDC is nonzero, and may also be used in other calculations such as heat transfer.

AF: Flow area—If no cross-sectional flow area is given or if AF is negative, AF is calculated by assuming that the duct is circular. Note that AF is therefore only needed for noncircular sections, and that the user may return to inputting circular sections in a FLOW DATA block by setting a negative value as default. *Otherwise, AF and DH are independent of each other.* AF is available in the user logic blocks and may be altered if necessary, but a negative AF can only be used as a signal in FLOW DATA blocks. It should be noted that AF and DH may be used to calculate the wetted perimeter for convective heat transfer. It should also be noted that the "size" of the tube is *inversely* proportional to AF. AF is also used to determine frictional pressure drops if IPDC is nonzero, and may also be used in other calculations such as heat transfer and accelerational pressure forces.

AFTH: Throat flow area—Flow area for any internal restrictions or contractions that may limit flowrates due to choking (flow cannot exceed sonic velocities in such throats). If no throat flow area is given or if AFTH is nonpositive, AFTH is assumed equal to AF. *Otherwise, AF and AFTH are independent of each other, although AFTH should be smaller than AF.* AFTH is available in the user logic blocks and may be altered if necessary. Currently, AFTH is only used in the optional choked flow detection and simulation routines CHKCHP and CHKCHL. Refer to the descriptions of those options in Section 6.9.6.6.

IPDC: Frictional pressure drop correlation number—While there are standard frictional pressure drop models for single-phase flow, there is no such agreement when it comes to two-phase flow. This flag specifies whether or not FLUINT should automatically perform frictional pressure drop calculations (for FC and FPOW described below), and which correlation should be used for two-phase flow. Setting IPDC=0 means FC and FPOW will not be updated for this tube. Any other value means FC and FPOW will be updated by the standard methods for single-phase flow, and the value of IPDC determines which of several popular two-phase correlations should be used. See Appendix A for further description of these friction models. The following list details the action taken according to the value of IPDC:

0 ..... No FC or FPOW calculations for this tube (user input)
1 ..... Standard single-phase model, plus McAdam's Homogeneous for two-phase (default)
2 ..... Standard single-phase model, plus Lockhart-Martinelli's for two-phase
3 ..... Standard single-phase model, plus Baroczy's for two-phase
4 ..... Standard single-phase model, plus Friedel's for two-phase
5 ..... Standard single-phase model, plus Whalley's recommended choice of 2, 3, or 4 (above) for two-phase
6 ..... Standard single-phase model, plus two–phase flow regime determination and frictional models based on those simplified regimes. Also invokes calculation of FD parameter when used in conjunction with twinned tubes.

Note that IPDC=1 is the default for single paths. If the user is not familiar with any particular correlation and has no preference, the default should be used. No other two-phase correlation produces "decent" results over such a wide range, although the others are more accurate for certain fluids and/or flow regimes (see Appendix A). If a more accurate prediction is desired but the user has no preference, try either IPDC=4 or IPDC=6.

Similarly, IPDC=6 is the default for twinned paths. Again, this default should be used for twinned tubes unless the user has advanced knowledge of two–phase regimes, wall friction apportionment, and interface drag correlations.

UPF: Upstream fraction of properties—This parameter is used if IPDC is not zero. This parameter allows the user to calculate FC and FPOW according to modeling preferences. If UPF=1.0, only the properties of the lump *currently* upstream of this tube will be used for the friction calculations. If UPF=0.0, only the downstream lump properties will be used. If UPF=0.5 (the default), both will be used for an average of the two. Any other UPF (between 0.0 and 1.0 only) will result in a weighted average.

UPF is also used to evaluate the effective fluid density in the FK loss term. Actually, the weighted average of the specific volume (the inverse of density) is used.

WRF: Wall roughness fraction—The roughness fraction is defined as the ratio of the characteristic length of the wall roughness to DH, and is usually on the order of 1.0^{-6} to 1.0^{-3}. If no WRF is given, it is assumed to be zero (smooth wall—this does not mean frictionless). WRF is available in the user logic blocks and may be altered if necessary. WRF is only relevant if IPDC is nonzero.

FK: Additional K–Factor Head Losses—Effects of pressure head losses due to entrances and exists as well as fittings such as bends, tees, expansions, contractions, orifices, valves, etc. may be included as K–factor head losses. Under steady flows, the irrecoverable pressure loss due to these effects is $\Delta P = FK \cdot FR^2/(2 \cdot \rho \cdot AF^2)$. Most undergraduate fluid mechanics texts contain tables and equations of appropriate values. Otherwise, *Flow of Fluids through Valves, Fittings, and Pipe* (Crane Technical Paper 410) is a definitive source. Note fully turbulent flow is often implicitly assumed in such tables and equations.

HC: Head coefficient—The head coefficient may be thought of as an additional pressure force superimposed on the tube. This may be caused by body forces or even by a pump contained within the tube (for a combined model). This parameter is left strictly to the user to specify and alter as needed. Otherwise, the value is assumed zero (no additional forces). The units of HC are pressure.

AC: Recoverable loss coefficient—A recoverable loss is essentially a pressure drop that becomes a pressure gain if the flow is reversed. The default is zero. This is most often caused by area and/or density changes between the inlet and the outlet of the tube, but may also be caused by radial addition or extraction (blowing or suction) of fluid). This parameter, which is automatically updated for tubes within HX and LINE macros, is described in detail in Section 4.7.12.

FC and FPOW: Irrecoverable loss coefficient and corresponding flowrate exponent—Unlike a recoverable loss, an irrecoverable loss remains a loss even when the flow is reversed. This is most often caused by wall friction, although bends, tees, and pipe fittings can contribute. When IPDC is nonzero, the exponent FPOW is a function of the current flow regime, and may vary from 0.0 (for laminar) to nearly 1.0 (for fully turbulent). With FPOW equal to 1.0, FC is proportional to a K-factor loss: $FC = -K_{eq}/(2\rho AF^2)$, where K_{eq} is the equivalent K-factor, ρ is the fluid density, and AF is the flow area as previously defined. If FPOW is 0.0, the resistance is linearly proportional to the flowrate. Note FC is typically a large negative number. *The units of FC depend on the value of FPOW.*

The user has several options when dealing with FC and FPOW. The user may (1) ignore them and let FLUINT calculate them according to default options, which is by far the most common approach, (2) specify the types of calculations to be performed (see below), (3) let the program calculate FC and FPOW and then modify or replace them, or (4) suppress FC and FPOW calculations altogether. When frictional pressure drops are desired, the user has many options available for calculating FC and FPOW. These options are controlled by parameters IPDC, UPF, and WRF, and are described below. It should be noted that setting FC to zero is valid, and corresponds to a frictionless pipe. However, *all of FK, FC, and AC cannot be zero for a steady-state solution, or with STUBE connectors.*

FD: Interface drag coefficient (Twinned tubes and STUBE connectors)—The interface drag term describing the retarding force exhibited by each phase on the other, in proportion to the difference in velocities. When IPDC=6, FD is calculated for each phase as $FD_k = -f_{vl}|V_v-V_l|/\rho_k$, where f_{vl} is the interface drag coefficient, V_v-V_l is the current difference between the velocities of each phase, and ρ_k is the density of the phase of the current tube. The interface drag coefficient is determined as a function of flow regime and void fraction, among other factors. Like FC, FD is typically a large negative number. If FD equals $-\infty$, the the flow will be homogeneous since any difference in velocity will be quickly extinguished. Along these lines, |FD| is typically larger for dispersed flow regimes (bubbly and slug) than for separated regimes (annular and stratified), resulting in smaller slip ratios (ratio of vapor to liquid velocity) for those regimes.

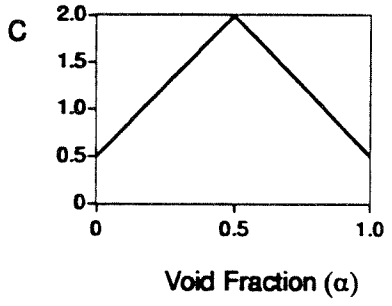
FG: Phase generation coefficient (Twinned tubes and STUBE connectors)—FG is the term covering the momentum transfer associated with phase change: when one phase is converted to another it must be accelerated or decelerated to compensate, and any velocity change must be caused by a force on the fluid. FG is calculated for each phase as $FG_k = \Gamma_t/\rho_k$, where Γ_t is the mass generation rate per unit volume of the twin phase, and ρ_k is the density of the phase of the current tube. (Since $-\Gamma_k=\Gamma_t$, it can also be defined as $FG_k = -\Gamma_k/\rho_k$.) The larger the rate of phase conversion, the larger |FG| becomes, although this is usually a small term.

Its relative insignificance is fortunate because of an inherent uncertainty in its formulation: to truly capture this term an interface velocity must be estimated, where the interface velocity is somewhere between the liquid and vapor velocities. In FLUINT, like several other similar codes, a donor formulation is used: the interface velocity is assumed to be the same as that of the donor phase. (While an average of the two velocities results in reversibility and therefore represents a physical limit, the donor formulation errs on the side of irreversibility.) The term ε_k governs this donor formulation. For evaporation, ε_k is zero for the liquid equation and unity for the vapor equation, in which case the liquid FG is positive and the vapor FG is negative. In other words, the FG term acts like a retarding force on vapor during evaporation, and an accelerating force on liquid during condensation.

This parameter is normally defaulted to zero. However, it is automatically updated for tubes and STUBE connectors within HX and LINE macros—see also Section 4.7.12.

AM: Added (virtual) mass coefficient (Twinned tubes, not applicable to STUBE connectors)—Because of the typically large difference in density between the liquid and vapor phase, by itself vapor has negligible inertia compared to that of liquid. However, each phase is not alone: because the acceleration of one phase involves the displacement of another, the distinction is not as clear cut. Consider a vapor bubble rising within a pool of liquid: after detachment from the bottom of the pool, the bubble does not accelerate instantly up to 'terminal velocity' because in order to do so it must displace an equal volume of liquid, which has arguable inertia. The added mass term (sometimes called virtual mass) can be thought of as a correction term that allows each phase to 'share' the inertia of the liquid phase, tending to reduce | dFR/dt | for both phases.

AM is calculated as $AM_k = C\varrho/\varrho_k$, where ϱ is the mixture density and ϱ_k is the density of the phase of the current tube. C is a unitless constant whose exact value is subject to debate. For separated regimes, C approaches zero since the phase can accelerate somewhat independently. For dispersed regimes, values of approximately 0.5 have been recommended. Because larger values of C generally increase stability, FLUINT errs on the side of larger C and does not use flow regimes in its estimation of C. Rather, C is estimated as a ramp function of void fraction, as shown below:



Although AM is calculated automatically, it can be overwritten in FLOGIC 1.

Connectors—Connectors, as mentioned before, are time-independent paths through which lumps exchange mass and energy. However, unlike tubes, there is no single interpretation for a connector—there is no real fluid component that resembles a "connector." Connectors, however, may be used to simulate a wide variety of fluid components such as pumps, valves, bends, orifices, and even certain capillary devices. For these reasons, it is convenient to think of connectors in terms of the devices they model. FLUINT offers a variety of prepackaged models to simulate a number of devices automatically.

Like junctions, connectors are governed by an algebraic, time-independent equation. They therefore do not *directly* affect the allowable time step during transients. The governing equation for the k^{th} connector, for which the positive flowrate direction is from lump i to j, is:

$$\Delta FR_k^{n+1} = GK_k^n(\Delta PL_i^{n+1} - \Delta PL_j^{n+1}) + HK_k^n + EI_k^n \Delta HL_i^{n+1} + EJ_k^n \Delta HL_J^{n+1} + DK_k^n \Delta FR_t^{n+1}$$

The connector equation applies during every solution time step or steady state relaxation of the network. The superscripts refer to the iteration or time interval; n is the value at the beginning of the solution, and all values superscripted n+1 are solved implicitly during the integration or relaxation. The parameters GK, HK, EI, EJ, and DK, along with a flow rate (FR), are all that any two connectors have in common. These parameters are defined as follows:

GK ..... partial derivative of the flowrate w.r.t. pressure drop changes
HK ..... flowrate offset: difference between next FR and current, where next FR is calculated assuming that the endpoint lumps do not change
EI ..... partial derivative of the flowrate w.r.t. upstream enthalpy changes
EJ ..... partial derivative of the flowrate w.r.t. downstream enthalpy changes
DK ..... partial derivative of the flowrate w.r.t. its twin's flowrate (if twinned)

Unless the connector is a twinned STUBE carrying two–phase flow, EI, EJ, and DK are by default zero. Furthermore, DK is ignored for any connector unless it is twinned. Thus, the above equation often reduces to:

$$\Delta FR_k^{n+1} = GK_k^n(\Delta PL_i^{n+1} - \Delta PL_j^{n+1}) + HK_k^n$$

The user has three choices when using connectors. First and most common, the user may name a device model and thereafter treat the connector as that device and ignore GK, HK, EI, EJ, and DK, which will be handled internally. The discussion of available device models is given in the next section. It should be noted here that all of these models are generalized, and that the user has access to their individual operating parameters within his logic blocks. Therefore, it is unlikely that one of these models can't be made to simulate a unique device whose operation can be described with a single path.

Second, the user may specify a *NULL* device, which forces him to take the responsibility of specifying and controlling GK, HK, etc. continuously. *This choice* (which is the default only because no other choice is appropriate) *should only be taken by the advanced user who cannot make a prepackaged model fit the particular device in question. Incorrect usage of GK, HK, EI, EJ, and can easily cause a fatal error or unexpected results.* However, this option is purposely

available so that FLUINT can be used by advanced users with unanticipated device modeling requirements. NULL connectors are also used by several simulation subroutines, such as NONEQ0/1 and COMPRS.

The third option, also recommended only for the advanced user, is to name a device, and then overwrite or modify GK, HK, EI, EJ, and DK in FLOGIC 1.

As general guidance, negative or zero GK should be used with caution. GK is often much less than unity in SI units, but much greater than unity in English units. HK is usually small compared to the value of FR, and is zero under steady conditions.

Actually, during the solution process the previously described tube momentum equations are converted into the same formulation as the above connector equation. Therefore, tubes also have the same parameters GK, HK, etc. However, these values are time–dependent, and are updated after FLOGIC 1 and before the integration. Therefore, while they are available for inspection in FLOGIC 2, they are normally of little use to the user.

Figure 4–6 summarizes the internal solution procedure for tubes and connectors in terms of when parameters are used, and in which logic blocks it makes sense to inspect or modify them. To use this chart, remember that tubes are treated like STUBE connectors in FASTIC. Connectors other than STUBEs follow similar rules—this figure can be extrapolated to them as well.

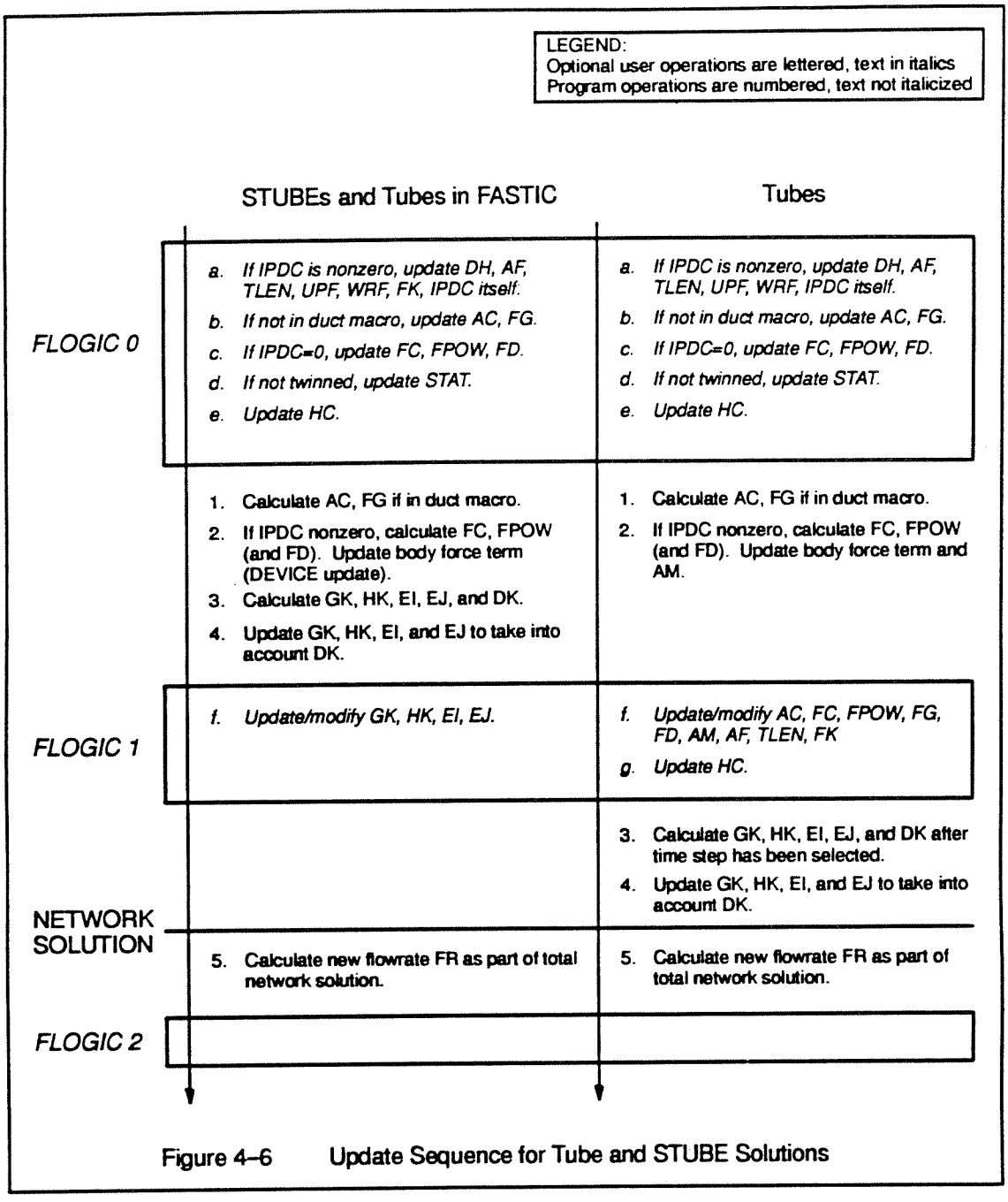


Figure 4–6 Update Sequence for Tube and STUBE Solutions

4.7.3.3 Ties

Ties provide the heat transfer links between lumps representing the fluid and nodes representing the wall. All ties have four descriptive parameters:

UA ..... the heat transfer conductance
QTIE ..... the heat rate through the tie
DUPL ..... the lump-side duplication factor
DUPN ..... the node-side duplication factor

UA has the same units as does the G of a thermal conductor, and operates in an analogous fashion for most ties (HTUS ties are an exception, as noted below):

$$QTIE_{tie} = UA_{tie} \cdot (T_{node} - TL_{lump})$$

where:

$QTIE_{tie}$ ..... Heat rate through tie, positive *into* lump and *out of* node
T_{node} ..... Node temperature
T_{lump} ..... Lump temperature

When a tie is made to a lump, the QDOT value is incremented by QTIE*DUPL, and nodal source term, Q, is decremented by QTIE*DUPN. Thus, energy appears in one network, exactly cancelling the energy that disappears in the other network. Actually, the heat rate may not be exactly equal to the UA times the temperature difference, especially in FASTIC or when junctions and/or arithmetic nodes are tied. This is because the heat rates on those elements affect their temperatures (or states) immediately, and because the heat rates are held constant over each solution interval to conserve energy. Another exception is HTUS ties, which differ from the above equation because of augmentation of heat transfer due to upstream effects, as described below.

Ties are unique because they are the only network element that can connect fluid submodels to thermal submodels, and they can only be used for that purpose. They are therefore very similar to intermodel thermal conductors, although they are strictly fluid network elements and always "belong" to the fluid submodel in which they were input. In order to tie two fluid submodels together, or to tie two lumps within the same fluid submodel (say, for example, to model a fluid-to-fluid heat exchanger or axial conduction terms), there must exist an intermediate thermal node that represents perhaps the separating wall. This is no restriction, since the fluids cannot mix. Also, note that a constant wall temperature condition can be simulated by tying a lump to a boundary node; a constant heat flux condition is modeled with the QDOT term of the lump, and no ties are needed.

Ties are also unique elements because *they alone can be dynamically moved from one location to another using the PUTTIE routine.* At every time step or steady-state iteration, the user has the opportunity of moving a tie to any other node and/or to any other lump within the same fluid submodel. One use for such "movable ties" is illustrated in Sample Problem F. Some types of ties (namely HTNS, HTUS, and CAPPMP HTM ties) move automatically.

Unlike conductors, a single tie cannot simultaneously exist between an arbitrary number of lump/node pairs. Each heat transfer path must be accounted for by a separate tie, but any number of ties may exist on any lump or node. The only exception to this statement is that CAPPMP junctions cannot support more the one tie optionally generated by that model.

Tie duplication factors function similar to path duplication factors, and in fact are intended to allow the user to deal with levels of duplication created by path factors that are not reflected in the thermal model. DUPN is the number of times the node "sees" the tie; DUPL is the number of times the lump "sees" the tie. Tie duplication factors may be used to mimic the treatment of SINDA one–way conductors: zero duplication factors on either end cause the equivalent behavior of a one-way conductor. *Zero duplication factors on both ends effectively eliminate the tie, and equal values other than unity can be used as a multiplying factor to degrade heat transfer coefficients, or to parametrically test the sensitivity of convection correlations.* Of course, duplication factors allow more general usage than do one-way conductors.

The user is cautioned that energy and mass can be created and lost via unequal path and tie duplication factors, and that although complicated nested levels of duplication are supposedly valid, the results can be somewhat confusing and the corresponding models difficult to debug. Use of the mapping routines QMAP, FLOMAP, NODMAP, and LMPMAP is recommended to help clarify the results. These routines show how energy and mass flow through the network.

There are six types of ties:

HTN .... Convection heat transfer, one path (tube or STUBE)
HTNC ... Convection heat transfer, centered, two paths (tubes or STUBEs)
HTNS ... Convection heat transfer, one path (tube or STUBE), segment version
HTU .... User-described heat transfer
HTUS ... User-described heat transfer, segment version
HTM .... Macro-specific heat transfer (CAPPMP)

HTM ties, which are currently only generated by CAPPMP macros, operate according to rules specific to that macro, which is documented in later sections.

Tie types differ in whether the user specifies the UA directly as a value held constant over each solution interval (HTU and HTUS), or whether the UA value is specified indirectly (i.e., is updated by the program). For example, by including path information along with the tie (HTN, HTNS, and HTNC options), the program will automatically calculate the UA value.

Tie types also differ according to whether they implicitly assume downstream–weighted heat transfer (referred to as lumped parameter ties), or whether they include the states of upstream lumps as well (in which case they are referred to as segment ties).

HTN, HTNC, and HTU ties are lumped parameter ties that calculate heat transfer assuming that the lump represents the bulk fluid properties, and that the node represents the average wall temperature. This implicitly assumes downstream–weighted heat transfer, since the lump state changes according to the heat transfer rate, and only that state is used in calculating heat transfer. Adjacent paths (tubes and STUBE connectors) are used only to provide flowrate and heat transfer area data; whether they are upstream or downstream of the lump is irrelevant.

On the other hand, HTNS and HTUS ties are *segment* ties. Segment ties treat the heat transfer problem differently than do traditional ties. Instead of being lump–oriented like the above class of ties, they are path–oriented. The node is assumed to represent the average wall temperature over the length of the path, the lumps on either end of that path are assumed to represent the upstream and downstream states of that segment. Unless the user specifies otherwise, *these ties will automatically jump to the lump that is currently downstream of the specified path when flows reverse.*

Further descriptions of tie options are deferred to Section 4.7.5.

4.7.4 Connector Device Models

Connectors can be used to simulate a wide variety of flow devices, as summarized in Table 4–5. The only restriction on these elements is that they are time-independent—all such devices are assumed to operate under quasi steady-state conditions; they react instantaneously to changes. Also, like any path, connectors do not have any associated mass and cannot accept or reject energy. These attributes are handled by lumps on either end of the connector.

Table 4–5 FLUINT Connector Device Models

NAME	DESCRIPTION
NULL	No device (user-input GK & HK.)
STUBE	Small or short (time-independent) tube. Same as a tube, but without inertia (reacts instantaneously).
CAPIL	Capillary (surface tension dominated) passage. For wicks, slots, tubules, grooves, etc.
LOSS	Generic head loss component. For losses in bends, tees, orifices, valves, etc.
LOSS2	Two way generic head loss component. Same as LOSS but direction sensitive.
CHKVLV	Check valve: LOSS element that closes for negative flow rates.
CTLVLV	Control valve: LOSS element that user may alter or close according to control logic.
MFRSET	Maintains a constant mass flow rate. For simple pumps, boundary conditions, certain valves (including closed,) etc.
VFRSET	Maintains a constant volumetric flow rate. Similar to MFRSET.
PUMP	Centrifugal pump, constant speed.
VPUMP	Centrifugal pump, variable speed (instantaneously varied speed.)

Beginning users should pay particular attention to STUBE and MFRSET connectors. Many models consist entirely of this devices.

4.7.4.1 NULL Option

If no prepackaged connector device model (as defined below) is appropriate for a particular modeling need and if the user is sufficiently advanced, then the NULL device option can be used. This suppresses GK, HK, EI, EJ, and DK calculations of any type; the user must manipulate these variables via the logic blocks. This option provides the user with great modeling power, but has several potential pitfalls if not used carefully. Therefore, it should only be used as a last resort. It is the default option, but only because no other default is possible. *WARNING: This option should only be used by advanced FLUINT users.*

The underlying coefficients in the linearized flowrate versus pressure drop relationship are defined on the basis of the equation: $\Delta FR = \Delta(\Delta PL) \cdot GK + HK + \Delta HL_{up} \cdot EI + \Delta HL_{down} \cdot GK + \Delta FR_{twin} \cdot DK$, which is obeyed by *all* connectors during each solution step. GK represents the partial derivative of flowrate with respect to pressure gradient (holding enthalpies constant), and is evaluated at the conditions that *would* exist if the flow were steady under the current boundary conditions—that is, if the endpoint lumps did not change. EI and EJ represent the partials with respect to up and downstream enthalpy (holding pressures constant) at that hypothetical flowrate, and DK represents the partial with respect to a twin's flowrate (if the NULL has been twinned). HK represents the difference in flowrate between the current flowrate and the hypothetical flowrate.

Zero GK means that the flowrate is independent of pressure gradient. Zero EI and EJ mean that the flowrate is independent of enthalpy (density, quality) changes. All parameters are updated by other connector options (between FLOGIC 0 and FLOGIC 1), and are available for inspection within the logic blocks. Thus, the NULL connector represents direct access to these normally "invisible" coefficients.

Some simulation routines such as COMPRS and NONEQ0/1 rely on NULL connectors, and the CAPPMP macro generates NULL connectors. In all these cases, the user does not have responsibility for updating GK, HK, etc.

4.7.4.2 STUBE Option

The STUBE (short or small tube) model is a time-independent version of a tube. The use of a tube to model either a vapor passage or a liquid passage with a large AF/(ρ•TLEN) ratio may cause small solution steps to be taken. If such resolution is beyond the scope of the analysis desired, then the user should use a connector with an STUBE model instead. *Most spacecraft thermal analyses do not require such resolution, so STUBE connectors should normally be used in preference to tubes.* However, tubes are relatively inexpensive if flowrates are not changing, and behave much more predictably than do STUBEs, especially in two-phase flows.

The operation of the STUBE model can be summarized concisely: an STUBE connector has all of the modeling options of a tube, but it reacts instantaneously to flow forces (e.g., always operates under steady-state assumptions). The governing equations for STUBE connectors correspond to those for tubes in the limits as dFR/dt goes to zero; all parameters have the same meaning and units. Note that there is no AM (added mass) term for twinned STUBE connectors since they have no acceleration associated with them. Therefore, with the exception of the inertial term AM, the STUBE connector has all the same descriptors and options as does a tube, so a detailed description will not be repeated here—refer to the previous section on tube parameters (DH, AF, HC, etc). Both tubes and STUBE connectors may be used with HTN, HTNS, and HTNC ties, and both may be used in macrocommands generating duct models (HX and LINE).

While STUBEs (and tubes, for that matter) might initially appear to be useful only for representing ducts, they are actually the most adaptable paths because of the access to the underlying fundamental modeling parameters HC, FC, FPOW, and AC. In fact, advanced users will find that *nearly all other devices can be equivalently simulated by manipulation of these parameters.* The HC factor can be used to simulate pumps to constant pressure drop devices like control valves. Using the AC factor, STUBEs may be used to simulate losses and pressure recovery due to area change in components such as reducers and diffusers (refer to Section 4.7.12)—*no other device can recover pressure.*

4.7.4.3 CAPIL Option

Surface tension forces can become dominant through small passages if both vapor and liquid are present. FLUINT allows simulation of such passages with the CAPIL connector device. The basic operation of such a passage is defined as follows: liquid may always pass through the path, but vapor cannot pass if the capillary limit (or bubble point) of the passage is not exceeded and any liquid is present to block the passage of vapor. All flow through the connector is assumed to be laminar because of the typically small effective diameter of such devices. There is no preferred direction: blockage may occur in either direction depending on conditions.

CAPIL connectors may be used to model wicks, grooves, tubules, thin slots, etc. They might be used as part of a model for gas traps, liquid acquisition devices, liquid/vapor separators, etc. However, there is one restriction on the use of this device: the STAT flag is disabled. This is a result of the internal use of liquid/vapor suction action to simulate the device. Also, it should be noted that a multi-element model (CAPPMP) exists which performs the same basic function as CAPIL, but allows heat addition/rejection and therefore can simulate capillary pumping (see Section 4.7.7). CAPIL connectors can stop vapor flow but cannot simulate pumping.

The following paragraphs describe CAPIL descriptors and their use. As with all connector models, all descriptors are available to the user in the logic blocks unless noted.

CAPIL has four descriptors: RC, CFC, XVH, and XVL. The last two parameters have default values, and will be discussed in Section 4.7.8. The first two, RC and CFC, have no defaults and will be described here. First, the definitions of RC and CFC are presented to aid in understanding them, and then methods of calculating them are given.

RC is the effective capillary radius of the meniscus if it were two-dimensional, defined such that the maximum capillary pressure that can be developed is:

$$\Delta P_c = 2\,\sigma / RC$$

where:

ΔP_c ..... Maximum capillary pressure gain or bubble point (psf in English units)
RC ..... Effective two-dimensional capillary radius (units of length)
σ ..... Fluid surface tension (units of force per length)

A spherical meniscus that forms in a circular passage has a radius RC. These menisci are typically found in tubules, although most wicking material is also described by an effective spherical pumping radius. A one-dimensional meniscus in the shape of a cylinder of radius R, would have an RC value of 2*R. Such one-dimensional menisci are typically found in open grooves and slots. In general, if the meniscus shape is described by two orthogonal radii R_1 and R_2, the input RC value should be $2*((R_1)^{-1} + (R_2)^{-1})^{-1}$. Thus, a square groove of width 1.0 units cut in the inside the circumference of a pipe of radius 10.0 units would have an RC of 2.0/(1.0 + 0.1) = 1.82 units.

Nonpositive RC values are interpreted as perfect capillary devices capable of withstanding or producing an infinite pressure gradient. This is useful for preliminary design work or for controlling steady-state solutions to prevent deprime.

CFC is the capillary flow conductance term, used to find the effective flow resistance of the path depending on the fluid and the phase of the fluid within the path. CFC is defined (for a single phase within the path) as:

$$CFC = FR * \mu / (g_c * DL * \Delta P_f)$$

where:

FR ..... Mass flow rate
μ ..... Fluid dynamic viscosity
g_c ..... Unit conversion factor for English units
DL ..... Fluid density
ΔP_f ..... Pressure drop due to friction

The usage of CFC is slightly more complicated for two-phase flow, which may only occur when the capillary limit is exceeded. As is typical with other paths, homogeneous phase distribution is assumed.

The following sections describe how to calculate RC and CFC for specific cases: tubules (and grooves), a uniform wick, and a thin slot.

For circular tubules, RC is simply one half of the diameter. CFC is then calculated using:

$$CFC = n * AF * DH**2 / (32.0 * TLEN)$$

where:

n ..... Number of tubules in parallel (duplication factors may also be used for multiple tubules, as described in a later section)
AF ..... Flow area of each tubule
DH ..... Hydraulic diameter of each tubule
TLEN ..... Length of each tubule

A single cylindrical tubule having the same laminar flow resistance as a CAPIL would have a diameter and length that follows the relationship $DH^4/TLEN = 128 \cdot CFC/\pi$.

Note that AF, DH, and TLEN are arbitrary names chosen here because of their familiarity—unlike tubes and STUBE connectors, CAPIL connectors do not accept these keywords as parameters. Also note that the formula given above can be converted into one applicable to open grooves by using appropriate values of AF, DH, and TLEN. In this case, the value of RC would be equal to the width of the open side of the groove (not half the width since a groove meniscus is one-dimensional).

For a uniform (isotropic) wick, the capillary radius should be available from test data, vendor data, or estimates. The same is true for the properties defining CFC. These are:

$$CFC = A * P / X \qquad \text{(flat plate)}$$

$$CFC = 2.0 * \pi * P * Y / ALOG(R_o/R_i) \qquad \text{(cylinder)}$$

where:

A ..... Wick total frontal area (perpendicular to the flow direction)
P ..... Wick permeability (units of area)
X ..... Wick thickness (in direction of flow)
Y ..... Wick length (perpendicular to the flow direction)
R_o ..... Wick outer radius
R_i ..... Wick inner radius
ALOG ..... Natural logarithm

The reader might notice that both of these equations are direct analogs to the equation for net conductance in a solid, with the permeability substituted for the thermal conductivity. Thus, the CFC for other more complicated wick geometries can be derived accordingly.

For a thin slot, RC is equal to the slot width W (because of the one-dimensional meniscus). CFC is then:

$$CFC = A * W**3 / (12.0 * X)$$

where:

A ..... Slot breadth (perpendicular to the flow direction—equals frontal area divided by W)
W ..... Slot width (gap) size
X ..... Slot length (in flow direction)

The user should be wary of the underlying assumptions in such a simplified model, and their impact on modeling a real capillary device, especially in off–design simulations:

1) The CAPIL, like any path, is fundamentally a one–dimensional component: it must be either primed or not. Real devices can exhibit more two–dimensional effects such as preventing vapor backflow in some regions of the wick, while simultaneously leaking vapor back into the liquid space in other locations.

2) The endpoint lumps are perfectly mixed, having a single thermodynamic state. While NONEQ0/1 routines have been used successfully to model deprime, and the BELACC routine has been used to model the spring–like meniscus, special care is needed to model behavior during deprime, or during clearing of the vapor lump.

3) While real wicks can exhibit considerable hysteresis (on the order of 50%) between the pressure differences at bubble formation and cessation, only a very small hysteresis (about 4%) is employed internally, purely for numerical stability.

4.7.4.4 LOSS Option

In addition to frictional losses in pipes, pressure head may be lost due to fittings such as bends, tees, orifices, valves, filters, etc. FLUINT provides a connector option called LOSS to simulate such head losses, which are characterized by a loss factor that is independent of flow direction through the path.

LOSS connectors are characterized by four parameters (FK, AF, AFTH, and TPF), all of which are available to the user to modify within logic blocks. (This capability can be used to simulate an in-line control valve according to the user's control logic.) With the exception of TPF, all LOSS connector parameters are also available in STUBE connectors.

FK is the loss element resistance factor, commonly called a *K factor*. AF is the effective flow area of the surrounding ducts (used as a basis for calculating velocity and head). AFTH is the effective flow area of any internal throat at which choking might occur (used only in optional routines CHKCHP and CHKCHL. TPF is an additional loss factor for two-phase flow effects. Each of these will be discussed below. The defining relationship for LOSS elements in single-phase flow is:

$$\Delta P = FK * (FR/AF)^{**}2 / (2.0 * g_c * DL)$$

where:

ΔP	Pressure drop caused by element
FK	K factor
FR	Mass flowrate
AF	Flow area (based on diameter used for K factor)
g_c	Unit conversion factor for English units
DL	Fluid density

For two-phase flow through the connector, the relationship is expanded to include phase densities and TPF:

$$\Delta P = FK * (FR/AF)^{**}2 * \frac{(1.0 + TPF * XL * (DL_l/DL_v - 1.0))}{(2.0 * g_c * DL_l)}$$

where:

TPF	Two-phase loss factor
XL	Fluid quality
DL_l	Liquid density
DL_v	Vapor density

FK can be any positive (nonzero) number. Tables of these values are available for common pipe fittings in any elementary fluid mechanics text*, and in many engineering handbooks**. A few words of caution should be noted here. First, AF should correspond to the pipe size used to find FK in a table. Second, K factors are only approximate, and this approximation can be poor

* See for instance: F. M. White, *Fluid Mechanics*, 1979.

** Recommended: CRANE Technical Paper 410, *Flow of Fluids Through Valves, Fittings, and Pipe.*

for laminar flow (specifically, below Reynolds numbers of about 10,000, where losses are highly geometry dependent). FK is available in user logic to be changed as needed to fit the analysis requirements.

K factors are based on flow velocity. At a given mass flowrate, the velocity increases as the quality increases for two-phase flows (according to a homogeneous flow assumption). This is taken into account in the above equation. However, additional losses usually result beyond that predicted by a homogeneous assumption. This may be taken into account using a value of TPF greater than 1.0. Fitzsimmons* gives information on finding TPF, and Rohsenow and Harnett† contains a short excerpt of this information. However, the default value for TPF is 1.0 (corresponding to only homogeneously predicted losses), and this should be used unless appropriate information is available. Once again, *if you don't understand it, ignore it.*

LOSS devices (and the related devices that follow) can only represent pressure losses. Non-positive values of FK are not allowed except in the CTLVLV option, where the sign is used as a flag. Pressure recovery due to area increase can only be modeled with the AC factor in tubes or STUBE connectors.

The user is cautioned against using values of FK that are too small, and against using LOSS-type devices in passages with very small flowrates—the flow conductance is infinite at zero flowrate. Both uses can adversely affect program accuracy and stability analogous to the disruption caused in SINDA by the use of conductors with very large G. Such strong connections blur the necessary distinction between adjacent lumps or adjacent nodes, violating the basic assumptions of finite networks.

The analyst should consider using the FK term in tubes and STUBE connectors instead of LOSS connectors. Combining both attributes into one path not only helps eliminate unnecessary lumps, it helps avoid some of the aforementioned limitations.

4.7.4.5 LOSS2 Option

LOSS2 is the same as LOSS, but loss coefficients are allowed to differ according to flow direction. Thus, only one new parameter is required. FKB, which is the K factor corresponding to backward (negative FR) flow rates. FK is the forward K factor, and the default for FKB is FK, in which case the connector is indistinguishable from a LOSS device.

LOSS2 may be used for certain flow control devices such as check valves (that do not completely close) or any other "diode" action device. LOSS2 devices are also appropriate for step expansions or contractions, but represent only pressure losses.

---

* D.E. Fitzsimmons, *Two-Phase Pressure Drop in Piping Components*, HW-08970 Rev. 1, March 20, 1964. (A report by GE for the AEC available from the Office of Technical Services, Department of Commerce.)

† W.M. Rohsenow & J.P Harnett, *Handbook of Heat Transfer*, 1973.

4.7.4.6 CHKVLV Option

CHKVLV may be used to simulate a check valve. A CHKVLV connector has all the same options and operations as does a LOSS connector, except that the valve will close automatically if the flowrate reverses *or if the pressure gradient reverses,* signalling impending flowrate reversal. The positive flowrate direction for the connector is assumed to be the nominal (open) flow direction. The valve will behave like a LOSS element when open (with a nonzero FK), and like an MFRSET connector with zero flowrate when shut. For this reason, a shut valve has all the same restrictions as an MFRSET or VFRSET connector (see below).

4.7.4.7 CTLVLV Option

CTLVLV may be used to simulate a control valve. With one exception, a CTLVLV connector has all the same options and operations as does a LOSS connector—the user is expected to describe the control valve operation via FK manipulations in logic blocks. The one exception is that the user may close the valve by using a negative FK. The valve may be opened and closed as desired, but FK can never be zero (i.e., there must exist some head loss across the valve, however small, when it is not shut). The valve will therefore behave like a LOSS element when open (i.e., FK is positive), and like an MFRSET connector with zero flow rate when shut. For this reason, a shut valve has all the same restrictions as an MFRSET or VFRSET connector (see below).

In many engineering applications, the performance of a control valve is given in terms of "C_v." This parameter is specific to English units and is usually used for water. To convert to unitless FK, use the following relationship (where AF is the valve area *in standard FLUINT English units* of ft2):

$$FK = 3.0E7 * (AF/CV)**2$$

Finally, a word of caution regarding changing K factors (for valves or loss elements) within logic blocks: unlike changes to thermal T, C, Q, and G factors, changes to FLUINT factors may result in instantaneous and complete changes throughout the fluid network. K factors should be adjusted gradually. The user must also avoid changes that result in oscillations; the value sensed should not be too dependent on the control valve reaction. Most control valve logic will require some sort of numeric damping. For these reasons, *the user may find that STUBE connectors with negative HC may make simulations of certain control valves easier, especially pressure regulating valves.*

Also, for flow losses that are not proportional to the square of the flowrate, consider CAPIL connectors for laminar losses (i.e., losses linearly proportional to flowrate) and STUBE connectors if the loss is proportional to some other exponent of flowrate. In the latter case, the user would set IPDC=0, and then assume responsibility for the manipulation of FC and FPOW in FLOGIC 0.

4.7.4.8 MFRSET Option

MFRSET means "set mass flowrate." All an MFRSET connector does is maintain the given mass flowrate through the connector, independent of pressure gradient or fluid state. This device model has only one simple parameter: SMFR, the flow rate to be maintained. This parameter is accessible in user logic blocks for inspection of modification.

MFRSET is a powerful modeling tool. It can be used to simulate an ideal pump or flow control device. It can even be used to control boundary conditions for open systems, or it can represent a completely closed valve (i.e., SMFR=0.0).

However, it must be used carefully or fatal errors will result. For example, using two MFRSET connectors in the same series line with different flow rates will eventually (if not immediately) cause an error. The user should be careful to avoid similar overspecifications in more complicated networks with parallel passages: at least one flow passage should be allowed to adjust itself without an MFRSET prescribing the flowrate. For this reason, MFRSET connectors cannot be the only type of path attached to a junction.

If the MFRSET represents a pump or compressor, the user should note that the program does not include the energy associated with such pumping. Indeed, all FLUINT paths are intrinsically adiabatic and therefore isenthalpic. In such cases, the user should add the appropriate energy rate to the downstream lump.

4.7.4.9 VFRSET Option

VFRSET means "set volumetric flowrate." It performs similar functions as MFRSET and should be used with similar caution. However, instead of maintaining a constant mass flowrate, VFRSET maintains a constant volumetric flowrate, independent of pressure gradient or fluid density. SVFR is the only parameter, and is used similar to SMFR (above).

VFRSET can be used in similar ways as MFRSET. However, it is a better approximation of a positive displacement pump, and has different behavior than MFRSET when used in compressible flows. Note that the mass flowrate through a VFRSET connector will drop considerably if there is any vapor in the inlet because of the density change.

If the VFRSET represents a pump or compressor, the user should note that the program does not include the energy associated with such pumping. In such cases, the user should add the appropriate energy rate to the downstream lump.

4.7.4.10 PUMP Option

A PUMP connector is a model of a centrifugal pump (i.e., a pump whose performance varies as a function of volumetric flow rate for a given rotational speed). In order to simulate such a pump, the user must specify a curve of pressure head (H) versus volumetric flow rate (G). The performance of the pump is then governed by the equation:

$$\Delta P = c * DL * (1.0 - DEG(\alpha)) * H(G)$$

where:

ΔP	Pressure difference due to the pump
c	Conversion constant, calculated internally
DL	Fluid density
DEG	Two-phase degradation factor (unitless) as a function of void fraction α
H	Head (units of length) as a function of volumetric flow rate G

By the above equation, note that the head H(G) is defined in terms of the density of the working fluid itself, not water, mercury, or some other reference fluid. Thus, one foot of head is the weight of a column of one foot of the working fluid under one gravity. H(G) is often provided in pump vendor data as a plotted curve. H(G) is specified by naming a user array (doublet) that contains the pump head curve. This array should have large enough range to cover all anticipated values. This includes operation with negative H or G. However, under the assumption that any answer is better than none, FLUINT will extrapolate from the available information to cover any excursions outside of the defined range. G must increase monotonically, and H must decrease monotonically. At least two data pairs are needed to define the simplest pump curve: a straight line.

Pumps designed to operate in liquid will generally suffer from significant degradation of head when two-phase fluid is encountered at the inlet. This degradation is typically characterized by a value that varies from 0.0 to 1.0 as a function of void fraction at the inlet. This is the purpose of DEG(VF), which may optionally be input by naming a user array (doublet) if such information is known.* However, this input is optional and is not always available for every pump. As a default, FLUINT will automatically use a default curve for DEG(VF), as shown in Figure 3–9. This default curve represents a reasonable average degradation for several pumps reported in the literature** and can be used as a first approximation unless specific pump characteristics are known.

Remember that no energy may enter or leave a network via a path. This means that *all paths represent isenthalpic processes*; FLUINT assumes that the energy added to the system by a pump is negligible. While this is a very good approximation for liquid-pumped transport systems with small pressure drops, it becomes less accurate for systems utilizing large pressure drops for control purposes. In such cases, the user should add or subtract the appropriate energy rate to or from the downstream lump. To simulate compressors and other turbomachinery, the attributes of both paths and lumps are needed since the "pumping action" cannot be thermodynamically separated from the "heating action," as it can with most true pumps or fans. The simulation routine COMPRS is provided for such simulations.

---

* FLUINT does not otherwise predict cavitation at the pump inlet cause by insufficient subcooling.

** B.E Boyack et al, *TRAC User's Guide*, LA-10590-M, Los Alamos National Laboratory, Nov. 1985.

J.J. Beyer, *FLASH6: A Fortran IV Computer Program for Reactor Plant Loss-of-Coolant Accident Analysis (LWBR Development Program)*, WAPD-TM-1249 (Westinghouse/Bettis Laboratory), July 1976.

Finally, it should be remembered that the PUMP model assumes a constant shaft speed independent of flow transients or mechanical inertia effects.

4.7.4.11 VPUMP Option

For cases where the assumption of constant shaft speed is inadequate, the user can simulate the effect of varying the shaft speed. The shaft speed in a real pump is actually a time-dependent parameter (e.g., governed by a differential equation in time), whose performance is a complicated function of time-varying hydraulic, friction, and motor and shaft torques, etc. Such simulations are difficult to characterize and are very hardware-dependent. Thus, they are beyond the immediate scope of FLUINT. However, this option allows the user to perform his own pump simulations in the user logic blocks. This capability not only allows modeling of realistic pumps, it allows simulation of pump-driven transients (start-up and shut-down).

VPUMP is very similar to PUMP, with three exceptions. First, a reference shaft speed, RSPD, must be provided (in any convenient units). Second, the H(G) array is assumed to be given at this reference speed. Third, the user can specify the current pump speed SPD, in the same units as RSPD. SPD is available to be altered in the user logic blocks.

VPUMP connectors operate using the pump similarity rules. These state that the volumetric flow rate is proportional to the shaft speed, and that the pump head is proportional to the square of the shaft speed. Thus, the governing VPUMP equation is:

$$\Delta P = (SPD/RSPD)^{**}2 * c * DL * (1.-DEG(\alpha)) * H(G*RSPD/SPD)$$

where:

SPD ..... Current shaft speed (any units)
RSPD ..... Reference shaft speed (SPD units)

Changes in SPD will occur instantaneously. Otherwise, VPUMP operates the same as PUMP. The effect of SPD on the pump curve is shown graphically in Figure 4–7 (no scale).

To be consistent with the similarity rules, SPD=0.0 is equivalent to a shut valve and carries all the restrictions that a shut valve does. Only the absolute value of SPD is considered (a centrifugal pump still produces a positive head with a reversed impeller). If a reversing pump is desired, two opposing parallel pumps may be specified between the same two lumps. They may use the same pumps curves in the same user array, but one would be turned off (SPD = 0.0) and the other on (SPD $\neq$ 0.0) according to the flow direction desired.

For fast transients where the mechanical inertias and equivalent electrical inertias (such as winding inductances) are important, the user must include such effects in the calculation of a time–dependent SPD. The time step may have to be restricted to assure that the otherwise instantaneous nature of connectors does not conflict with such simulations. As an alternative, consider a tube whose inertia (TLEN/AF) and other terms (HC, FK, etc.) are manipulated in logic blocks to simulate a truly time–dependent pump.

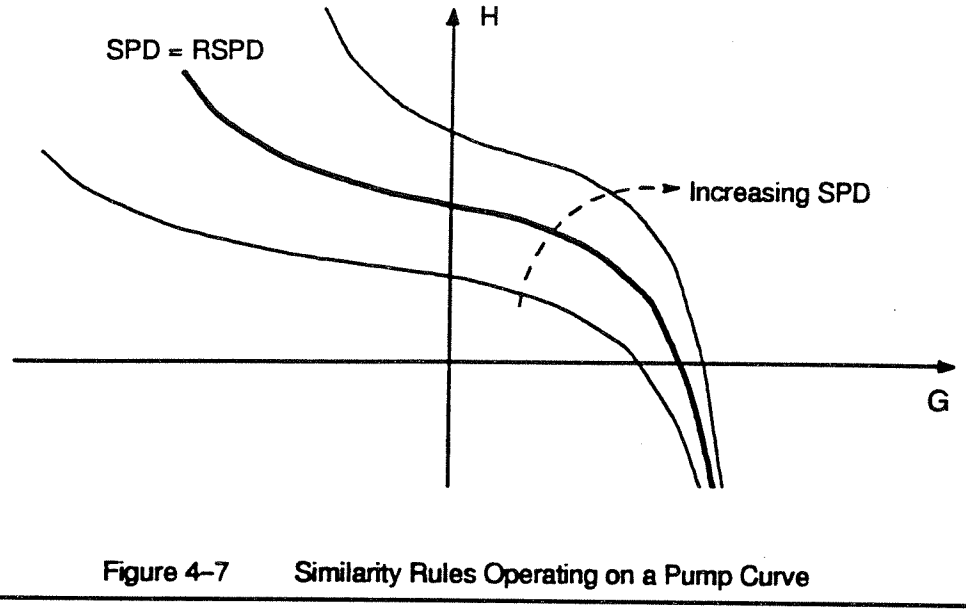


Figure 4–7 Similarity Rules Operating on a Pump Curve

4.7.5 Heat Transfer Methods

Heat transfer calculations in FLUINT may be effected in several ways. First, the user may calculate heat transfer rates and manipulate lump QDOT (and perhaps nodal Q) values as needed within the user logic blocks. This method is appropriate in constant heat flux environments where the energy exchange rates are independent of fluid flow, but can be very difficult to implement in other cases because of the potential for numerical instabilities. Alternatively, the user may opt to let FLUINT perform heat transfer automatically using *ties*. The remainder of this section describes the how ties are used.

Ties may be divided into subtypes according to whether (1) the tie conductance (UA) value is provided by the user or calculated by the program, and (2) whether the lump is intended to completely represent the bulk fluid or merely the downstream end of a heat transfer section.

The first distinction is clearly made. With HTN, HTNC, and HTNS ties, the UA value is calculated automatically by the program according to convective heat transfer in a duct. With HTU and HTUS ties, the user provides the UA coefficient.

The second distinction is more subtle. HTN, HTNC, and HTU ties calculate heat transfer assuming that the lump represents the bulk fluid properties, and that the node represents the average wall temperature. This lumped parameter formulation implicitly assumes downstream–weighted heat transfer, since the lump state changes according to the heat transfer rate, and only that state is used in calculating heat transfer. Adjacent paths (tubes and STUBE connectors) are used only to provide flowrate and heat transfer area data; whether they are upstream or downstream of the lump is irrelevant. While these methods are robust and are recommended for two–phase flows, they tend to underpredict heat transfer rates slightly for single phase flows, especially when the resolution is too coarse. Also, they tend to conflict with the way many engineers treat a single–phase heat transfer problem: as a constant wall temperature over a segment that has distinct inlet and outlet states.

HTNS and HTUS ties, which will be referred to as *segment* ties, treat the heat transfer problem differently than do traditional ties. Instead of being lump–oriented, they are path–oriented. The node is assumed to represent the average wall temperature over the length of the path, the lumps on either end of that path are assumed to represent the upstream and downstream states of that segment. Unless the user specifies otherwise, *these ties will automatically jump to the lump that is currently downstream of the specified path when flows reverse.*

Sample Problem H in Appendix F helps illustrate the difference between these two classes of heat transfer ties.

4.7.5.1 Lump–oriented (Lumped Parameter) Ties

HTU Ties—When using an HTU tie, the UA conductance must be provided by the user, perhaps as calculated in the FLOGIC 0 block. No path or flowrate is associated with an HTU tie. Thus, HTU ties may be used to represent almost any heat transfer phenomena including natural convection, developing velocity profiles, pool boiling, grooved evaporation or condensation, and jet impingement.

While such phenomena are the purpose of HTU ties, although they may also be used to override the standard convection correlation set used in the routine STDHTC, as documented in Section 6. Instead of using HTU ties for alternate convection correlations, however, it is recommended that the user instead use HTN and HTNC ties (described below) and then replace any convection correlations by inputting a routine with the same name and same argument list in SUBROUTINE DATA.

The reason for this recommendation is the same as the reason for a caution in the use of HTU ties. *The program must assume the UA value of an HTU tie stays constant over each solution interval, and this may be a source of oscillations when used in combination with time–independent elements* if the user updates the conductance purely based on the state of the network before each solution step. With other ties, the UA product is calculated simultaneously with the lump state and nodal temperature by iterative calls to the convection correlations. Such "implicitness" improves stability but is not possible with HTU ties because the user is given limited opportunities to update the UA product (namely in FLOGIC 0 and 2 blocks).

For example, assume a user updates a UA value in FLOGIC 0 for a tie on a junction based on the state of the junction. If the quality of the junction is 0.0, a low value of UA is used for heat transfer in a liquid. If the quality rises above 0.0, a much larger value of UA is used to represent two-phase heat transfer. Because junctions can change instantaneously (as can tanks in FASTIC), the junction can easily oscillate between liquid and two-phase if the wall node is colder than the two-phase fluid entering the junction. A large two-phase coefficient cools the junction to single-phase; a small single-phase coefficient allows the junction to warm up into the two-phase region again. The problem is the same when the wall node is warmer than the two-phase flow entering the junction.

To counteract such oscillations, the user might attempt to anticipate such effects as much as possible by solving for an appropriate value of UA simultaneously with the junction state. This would become cumbersome for strings of junctions because of the dependency on upstream conditions. An easier though less physically meaningful method would be to add hysteresis or damping into the UA product update, which is not always as easy as it sounds, nor is it always successful.

Note that the user faces a similar problem if the QDOT value of an untied junction is updated according to the state of the junction. Section 4.7.16 discusses further the effects of updating parameters with instant feedback. Sample Problem E provides an example of these options as applied to a specific case. Sample Problem C contains logic that avoids such instabilities in steady–states by calculating the initial condition directly. In that sample problem, the dependence of a pool–boiling correlation on the third power of temperature difference is an example of when the assumption of constant UA can be both poor and destabilizing because of the treatment of diffusion nodes as instantaneous elements during steady state solutions.

HTN and HTNC (Convection) Ties—When these ties are used, the UA conductance for the tie is not specified directly by the user. It is instead calculated by the program according to the user's instructions. These ties are based on forced convection (including boiling and condensation) in internal ducts assuming constant wall and fluid temperatures over each solution interval. Appendix B describes the set of heat transfer correlations, which are all access via the central routine STDHTC.

To enable FLUINT to calculate a UA based on forced convection, users must identify more than a container node and a fluid lump—they must also name at least one path.* Energy may only added or extracted at fluid lumps. However, lumps have no specific geometry (except perhaps a volume), and cannot be characterized by a flow regime. Tubes and STUBE connectors, however, have specific geometry (areas, diameters, lengths) and flowrates, but have no associated mass or thermodynamic properties. However, convection heat transfer calculations depend on specific properties, geometries, flow rates, and wall temperatures. Thus, these predictions require the attributes of lumps, paths, and thermal nodes to be combined in one calculation. In order to describe the methods behind convection ties, some preliminary discussions of duct modeling are necessary.

Diabatic ducts, a necessary component in any heat exchanger model, may be modeled in FLUINT as a series of discretized sections. There are two basic options for describing each section.

First, the user may "lump" all the associated mass and volume of a heat exchanger section into a tank (or other lump), and connect this tank to the up- or downstream lump with a single path that has the appropriate geometry of the section. This is the basis of the HTN tie. With only one path used, energy is assumed to enter or leave the section at one end or the other (upstream or downstream—see Figure 4–8). Flow reversals do not change the end at which heat is exchanged (i.e., transfer at the downstream end of the section becomes transfer at the upstream end). The governing equation for this type of heat transfer is then:

$$UA = F * AX * HTC$$

where:

F ..... Fraction of active wetted area (or fouling factor)
AX ..... Heat transfer area (wetted area of path: 4.0*TLEN*AF/DH)
HTC ..... Heat transfer coefficient based on thermodynamic state of lump, temperature of wall, and flowrate and geometry of path

Alternatively, the user may "center" the lump between two paths to represent the section. This is the basis of the HTNC ("centered") tie. In other words, the energy entering or leaving the lump is the sum of the heat transfer calculated with two paths representing the container wall. One path may be upstream of the lump, and one may be downstream. For a string of such sections, the paths between the lumps may be shared (divided in two) for heat transfer purposes, as represented in Figure 4–9.

---

* Throughout this section, reference to *one path* or a *single path* can also be interpreted to mean reference to *the primary twin of a pair of twinned tubes or STUBE connectors*. Only the primary twin need be named, the involvement of the secondary twin, which shares the same passage, is implied. In other words, despite the presence of two paths, only one passage of the appropriate diameter, length, and heat transfer area exists. Thus, an HTN tie can refer to a single path or a single pair of twins. An HTNC tie can refer to two single paths, one single path and one pair of twins, or two pairs of twins.

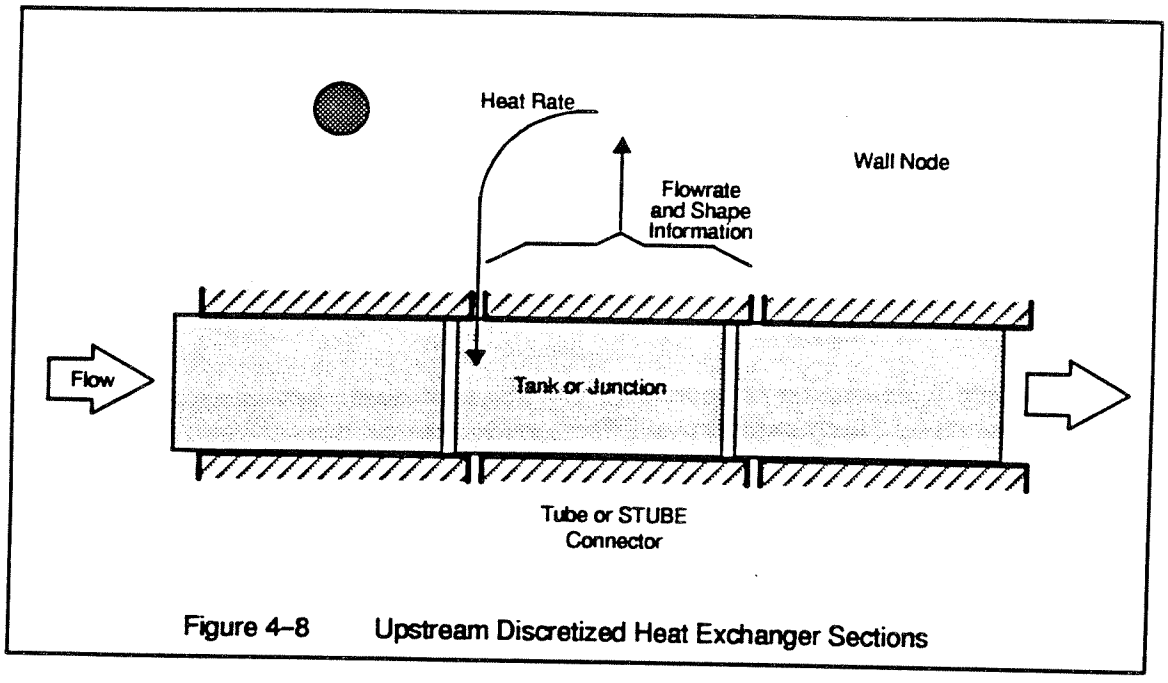


Figure 4–8 Upstream Discretized Heat Exchanger Sections

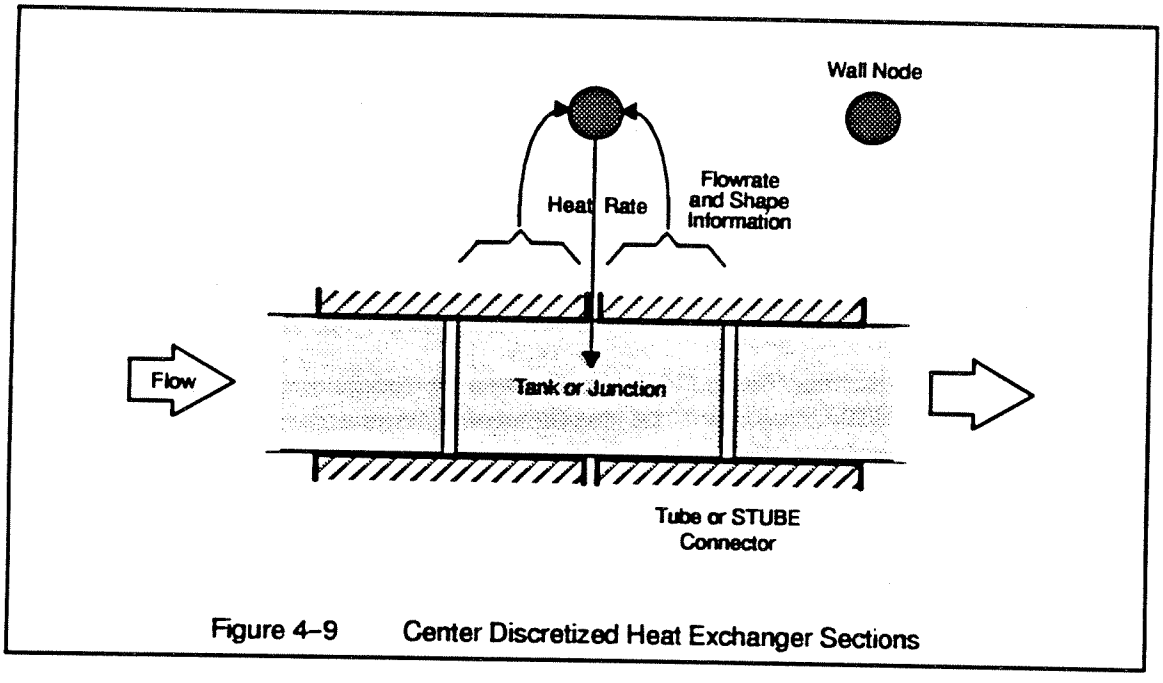


Figure 4–9 Center Discretized Heat Exchanger Sections

Actually, as noted in the following equation, the lump does not ve to reside in the exact center of the path sections, and does not have to share a path at al The HTN tie is simply a special case of the HTNC tie. However, if the lump is centered (F1=F2=0.5 for equal sized up and downstream paths), flow reversals will not alter the basic assumption of center-added heat. The equation for HTNC ties is:

$$UA = F1 * AX1 * HTC1 + F2 * AX2 * HTC2$$

where:

F1 ..... Fraction of path 1 whose volume is contained in lump (fraction of active wetted area or fouling factor for path 1)
AX1 ..... Heat transfer area of path 1 (wetted area of path)
HTC1 ..... Heat transfer coefficient based on thermodynamic state of lump and flow rate and geometry of path 1
F2 ..... Fraction of path 2 whose volume is contained in lump (fraction of active wetted area or fouling factor for path 2)
AX2 ..... Heat transfer area of path 2 (wetted area of path)
HTC2 ..... Heat transfer coefficient based on thermodynamic state of lump and flow rate and geometry of path 2

The area fractions for both HTN ties and HTNC ties are really just multipliers, and may be used to augment heat transfer (due to thinner than fully-developed boundary layers or impingement) or to degrade it (due to fouling). However, the primary use is for ducts whose heat transfer wetted area is not equal to the frictional wetted area: some parts are adiabatic. For example, a square duct with one side adiabatic would have an area fraction of 0.75. A duct with heat added over one half of its length (or in which the other half of the path representing the duct were accounted for by another tie) would have an area fraction of 0.5. Tie duplication factors DUPN and DUPL are perhaps better suited for fouling or other multiplying factors since these can be dynamically altered within user logic blocks, whereas area multipliers can only be specified once within the FLOW DATA block.

As guidance, up- or downstream-lumped pipe and heat exchanger sections are slightly more efficient than center-lumped sections, especially for single-phase flow. The reason is that heat transfer will not change significantly from one method to the other, although pressure drops may differ slightly with two-phase flow (causing an indirect effect to the heat transfer calculations). Using a center-lumped method may require a smaller time step (because of the half-sized paths on the ends) and will require an additional lump/path pair to produce the same resolution of up or downstream-lumped methods. However, a center-lumped method is less sensitive to flow reversals and *should be the method-of-choice for two-phase flow* because of large property gradients along the line. **When in doubt, use the center-lump method.**

A heat exchanger section may also be modeled using a junction in place of a tank if the mass and energy storage capabilities of the section are neglected. In fact, this approach is strongly recommended for two-phase problems if transient liquid inventory changes within the section are not part of the problem. Using junctions instead of two-phase tanks will significantly increase solution speed. Both plena and boundary nodes may be named in ties used to represent boundary conditions for either thermal or fluid submodels.

Finally, it should be noted that there are several powerful preprocessor commands available in FLUINT that allow fast generation of complete heat exchanger models (as well as segmented line models without such heat transfer calculations). These commands (LINE and HX) are discussed in the next section (4.7.6).

Inherent Downstream-weighted Heat Transfer—The above discussion on upstream, downstream, and centered discretizing relates to where the lump is located with respect to paths, which dictate flowrates and heat exchange area. However, this assignment is unrelated to where the lump is located with respect to the wall node(s). In fact, inspection of the equation QDOT = ΣQTIE = Σ(UA*(T – TL)) reveals an inherent assumption of downstream–weighted heat transfer: under steady conditions, the temperature (or quality) of the lump is a function of the heat transfer into it, which is in turn a function of its temperature (or quality). In other words, the heat transfer rate is calculated on the temperature at which fluid exits the perfectly mixed lump. *When plotting results versus lump axial location, the lump should be considered to exist at the downstream end of each section when using lumped parameter ties.*

This lumped parameter formulation is stable, and is entirely consistent with the fundamental assumptions of a lumped–parameter network. However, when lumps are coarsely discretized (i.e., too few lumps are used to adequately represent the flowwise temperature gradient), the heat transfer rate will be underestimated, especially if the node is perceived as representing an average wall temperature for the section.

Strictly, the correct solution is to not abuse the network assumptions, and use an adequate number of elements to capture the desired number of states. However, the user may alternatively choose to interpret the lump states as representing an inlet and outlet condition, with the node representing the average wall temperature from inlet to outlet. For single–phase flow, where the heat transfer coefficients and flowrates are uniform axially, the types of ties described in the next section may be more appropriate.

4.7.5.2 Path–oriented (Segment) Ties

As noted in the above discussion, coarsely discretized single–phase lines tend to underestimate heat transfer rates. This section describes ties that model single–phase heat transfer using segments rather than lumped parameters.

The previously described lumped parameter ties (HTU, HTN, HTNC) link a node and a lump, and may optionally point to one or more paths, which are used to define size (heat transfer area), shape, and flowrates. The node represents the wall; the lump represents the fluid. In those ties, the lump is viewed as coincident with the node, and each represents the average fluid/wall state. No distinction is made for up or downstream. For HTNC ties, the flowrates and sizes of two paths are used, but the lump is considered centered.

WIth the alternative *segment tie*, the link is defined between a node and a *path*. Instead of tying to a lump directly, the user ties to a path representing the segment. The node represents the average wall temperature over the entire length of the path, and the endpoint lumps represent the inlet and outlet states. Of course, since heat may only be added to a lump, the tie is really attached to the downstream lump, and the UA and QTIE values simply reflect the aug-

mentation caused by the inclusion of an upstream state. *However, if the flowrate reverses, then by default the tie automatically jumps to the opposite end of the path. Any QDOT applied to the upstream lump will be overwritten if this jump occurs.* (Alternately, the user may command the tie to stay put on the defined downstream lump by issuing a "STAY" command in the input sub-block.) Because of the potential for flow reversals, the link to the endpoint lumps is somewhat indirect.

Two types of segment ties exist: HTUS and HTNS. With HTUS ties, the analogs of HTU ties, the user is expected to input the UA value. Unlike HTU ties, a path (of any type) is named.

With HTNS ties, the analog of the HTN tie, the program calculates the UA based on convection calculations for the path representing the segment (Figure 4–10). No analog exists for the HTNC tie.

For segment ties, $QTIE = (Q_{up} - Q_{down})/\ln(Q_{up}/Q_{down})$, where $Q_{up} = U_{up}*A*(T_{node} - T_{lump,up})$ and $Q_{down} = U_{down}*A*(T_{node} - T_{lump,down})$ by default, representing an logarithmic average of the inlet and outlet heat transfer rates. For single–phase flows, U is relatively constant and this solution is equivalent to a log–mean temperature difference (LMTD) solution. If either lump is two–phase, or if flows reverse and the "STAY" option has been used, the ties revert to downstream–weighted (HTU and HTN) behavior: $QTIE = Q_{down}$. For single–phase flows, if $|Q_{down}| > |Q_{up}|$ then an arithmetic average is used. In the special case of HTUS ties in single–phase flows, the QTIE value will usually exceed $QTIE_{tie} = UA_{tie}*(T_{node} - T_{lump})$ because of the effects of the upstream lump.

Unfortunately, since segment ties represent a different way of looking at the same problem, they are not always easily convertible to or from the lump–oriented type of tie without other model changes.

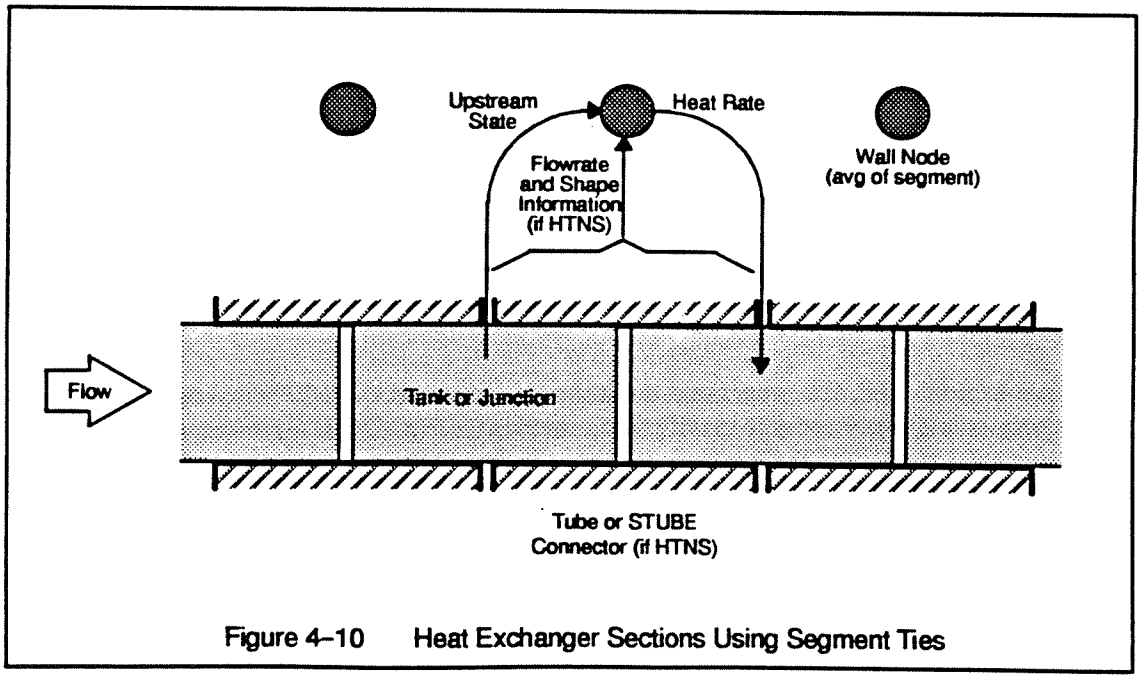


Figure 4–10 Heat Exchanger Sections Using Segment Ties

Lumped Parameter vs. Segment Ties—Improvements of segment ties over lumped parameter ties exist but aren't always significant unless very coarse spatial resolution is used (refer to Sample Problem H). Furthermore, the limiting assumption in such coarse models often becomes that of constant wall temperature. In other words, if the node temperatures vary axially, then the user is forced to discretize as needed to capture this gradient, which in turn drives the selection of the flow model resolution. In addition to wall temperature, the user must be cognizant of other important property variations, such as pressure (for compressible fluids), viscosity, and conductivity. Velocities, densities, Reynolds numbers, and heat transfer coefficients can all vary axially as a result, potentially placing another lower limit on acceptable resolution. Nevertheless, in preliminary designs where models are often very coarse and analyses are all steady–state, these ties can be used to collapse an entire single–phase heat exchanger model into one node, one tie, and two lumps.

While segment ties function adequately for two–phase (including slip flow) and compressible fluids, they can be problematic models when using tanks if the flow reverses while being cooled or condensed. For example, if by some perturbation a flowrate temporarily reverses in the middle of a two–phase model of a condenser that uses tanks, then the potential exists for two ties to jump to the same tank, virtually assuring its collapse to all liquid. HTNC ties avoid such problems not only by staying put even when flows reverse, but also by averaging velocities axially when calculating UA. The use of the "STAY" command helps alleviate some but not all of this difficulty.

In summary, ***while HTNS ties are preferred in the limit of single–phase steady flows, HTNC ties are recommended for two–phase unsteady flows or where flow conditions are not known a priori.***

Another potential problem related to reversing flows with these ties is the use of unequal (asymmetric) path duplication factors: DUPI $\neq$ DUPJ. Recalling that such duplication factors affect the amount of heat transfer area 'seen' by the lump, the value of UA may change by the ratio DUPI/DUPJ for HTNS ties if flows reverse and the "STAY" command has not been issued. A warning is produced at the start of the processor if it detects the potential for such a condition given the *initial* values of DUPI and DUPJ and the absence of the "STAY" command, but no further actions or checks are performed thereafter.

4.7.5.3 How Ties are Effected

Ties make energy disappear from one submodel and reappear in another. Lump QDOT terms and node Q terms are affected. The heat rate through the tie, QTIE, is summed into the QDOT term of the lump (multiplied by DUPL) and subtracted from the Q term of the node (multiplied by DUPN).

When a lump is tied, the program automatically updates the QDOT value for that lump, leaving the user one opportunity to modify that heat rate (in FLOGIC 1) for tanks, and no opportunities to modify the QDOT for junctions (or tanks in FASTIC). In most cases, *the user forfeits the ability to independently specify or modify QDOT for tied lumps.* Ties are effected (and QTIEs, UAs, and QDOTs updated) between FLOGIC 0 and FLOGIC 1 in all routines.

The rules governing Qs for tied nodes are slightly less clear. The nodal term is summed into (or subtracted from), rather than replaced. This summation occurs after VARIABLES 0 but before VARIABLES 1 in all routines: ties are treated like time–varying sources. Recall that nodal

Qs are reset every solution step, unlike lump QDOTs. Because VARIABLES 0 is called only once at the beginning of a steady-state solution (FASTIC or STDSTL), nodal Qs are reset to their post-VARIABLES 0 (e.g., time-dependent) value at the start of every iteration, and the effect of ties is then summed into those values. However, for stability reasons tied lumps are treated like boundary nodes from the point of view of the thermal solution during steady-states—even though the Qs are updated rigorously, they are irrelevant in steady-states. Thus, the UA values are used by the thermal solution in steady-states, but the QTIE values are used in transients. Even in transients, however, the UA value is summed into the CSGMIN calculation for stability reasons.

When using the TIETAB output routine or network mapping routines, it is often possible that the QTIE value does not exactly correspond to the UA value multiplied by the temperature difference, even for converged solutions. The cause of this apparent discrepancy is often that the QTIEs were calculated based on lump and node states at the beginning of the solution interval, rather than on the values that appear in output listings at the end of the solution interval. Other reasons include internal damping, etc. For HTUS ties, the discrepancy is inherent since the UA value is user–provided but the QTIE value reflects the augmentation of the upstream state.

4.7.6 Macros: Convenient Element Generation

By now, the reader will have noticed the very large number of input parameters and options in FLUINT. This makes the program flexible and powerful, but makes the job of inputting models difficult. The FLUINT macros can be used to alleviate this difficulty.

One feature useful for reducing inputs is the ability to explicitly define or redefine default values at any point in the input stream. The default values for most FLUINT variables may be defined in this way. Also, the preprocessor allows macrocommands (or *macros*) whereby many elements may be generated with a single command, thereby decreasing the amount of input and increasing the readability of the input file. While it is not the intent of this section to present input details, the types of macros available will be introduced here because of their relation to capabilities like heat exchanger modeling.

There are three types of macros. GEN type macros allow fast generation of repetitive inputs. The LINE macro allows fast generation of pipe or line models (without heat transfer ties). The HX macro performs the same function as LINE macros, but automatically adds a row of radial heat transfer ties to thermal submodels for heat exchanger modeling. These macros are described briefly below.

GEN Macros:

GENLU ..... Generates several lumps of the same type.
GENPA..... Generates several paths of the same type.
GENT ..... Generates several ties of the same type. Can be used to add another row of ties to a LINE or HX macro, even if centered.

The LINE Macro:

U Option .... Generates pairs of lumps and paths in series: lump —> path —> lump —> path ... that model a line or pipe. Lumps are upstream of the corresponding path. Paths are tubes or STUBE connectors. Lumps are tanks, junctions, or plena. No ties are generated.
D Option .... Generates pairs of paths and lumps in series: path —> lump —> path —> lump ... (Same as U option but lumps are at downstream end of each section.)
C Option .... Generates a series of paths and lumps that begin and terminate with paths: path —> lump...(more path/lump pairs)... —> path. (Lumps are centered between corresponding paths.)

The HX Macro:

U Option .... Same as U option with LINE but generates HTN ties automatically. Allows fast building of heat exchanger sections.
D Option .... Same as D option with LINE but generates HTN ties automatically. Allows fast building of heat exchanger sections.
DS Option ... Same as D option with LINE but generates HTNS ties automatically. Allows fast building of heat exchanger segments.
C Option .... Same as C option with LINE but generates HTNC ties automatically. Allows fast building of heat exchanger sections.

The differences between the U, C, and D options may seem confusing at first. Because centered (C) options start and end with paths, they may be treated as a single "superpath" and may be easier to apply. For this reason, and because the centered option is recommended for two-phase flows, **use the centered (C) option if there is any doubt.** See Figure 4–11.

Note that the PPSAVE option will not expand FLUINT macrocommands into the individual elements because information is lost when elements are input separately. However, the preprocessor output file will contain the data generated by the macros for verification purposes.

Use of LINE and HX macros invoke logic that automatically calculates AC and FG factors. These factors take into account axial variations in velocity and the momentum transfer associated with phase change (if twinned). Thus, these models will respond differently than if the elements had been input individually—FLUINT treats them like continuous ducts rather than as isolated elements. For these reasons, ***the use of LINE and HX macros is strongly recommended over individually input elements representing a single continuous flow passage.***

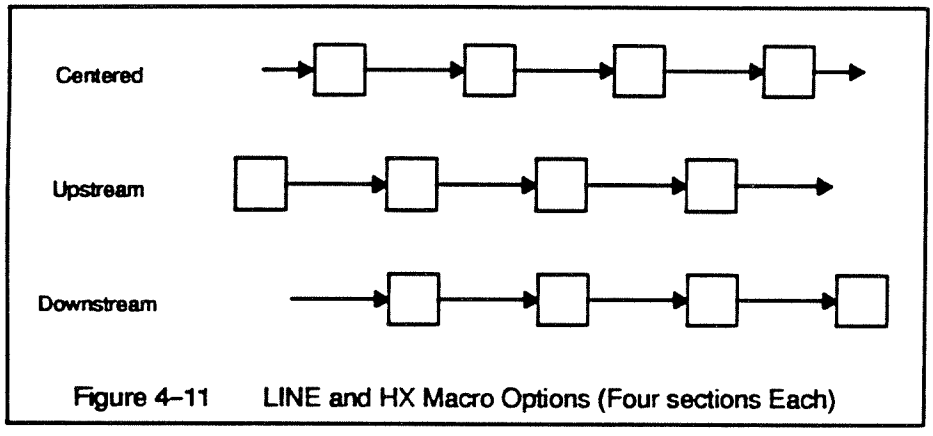


Figure 4–11 LINE and HX Macro Options (Four sections Each)

4.7.7 Models: Multi-Element Component Simulators

A large number of common fluid system components may be modeled in FLUINT by using a single element (this includes device models for connectors). However, certain complex components require the characteristics of both lumps and paths and perhaps ties in their descriptions. Such components are handled using multi-element *component models*.

Like macros, component model commands generate several elements. Unlike macros, model commands only generate a fixed number of elements. These elements are handled together as a single unit by model simulation logic that is automatically invoked by naming a model. Although users must be aware of the elements being generated (in order to provide them with identifiers), they do not need to remember the specifics of every parameter for every element. For example, although connectors used in component models may not correspond to any device model (such as STUBE or CAPIL) and may be denoted as NULL connectors, the user is not responsible for manipulating their GK and HK values. The parameters of each element in the component model are manipulated collectively inside the processor.

The remainder of this section describes the only component model currently available, the CAPPMP. In previous versions, a hardware–specific model was also available; others may be added in the future. See also the discussion on capillary modeling and phase suction options in the next section, and the SPRIME routine documented in Section 6.

The CAPPMP Macro—Capillary pump model. This macro generates a junction between two NULL* connectors:

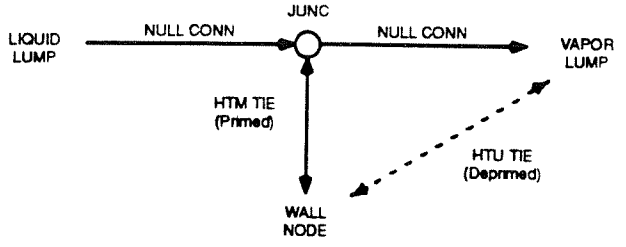
The simplest way to think of a CAPPMP model is a CAPIL connector with a junction in the middle. The presence of the junction allows heat to be added or removed from a CAPPMP model, whereas a CAPIL connector is adiabatic like any path. *It has no other physical meaning, and the state of the junction is largely meaningless.* If the heat rate into the junction is zero, the CAPPMP behaves exactly like a CAPIL connector. *Unlike a CAPIL, the CAPPMP is direction sensitive.* The vapor end (the lump at which heat transfer may be assumed to occur) must be designated.

If the junction QDOT is positive, the CAPPMP simulates an evaporator-pump. Liquid at the designated liquid end (i.e., *not* the designated vapor end) can be evaporated and forced into the designated vapor end, performing pumping by capillary action. If the pressure drop is too big or a capillary interface cannot otherwise exist, the device will *deprime*. This will generate error messages if the flowrate through the junction suddenly drops yet the QDOT continues to apply. See the discussion in the next section and the subroutine SPRIME. If the junction QDOT is negative, vapor (if located at the designated vapor side) will be condensed will be allowed to pass to the lower pressure liquid side.

As an alternative to directly specifying the QDOT of a CAPPMP, the user may generate an HTM tie and provide a UA coefficient. No other ties may be added to a CAPPMP junction. Because the junction within a CAPPMP macro is really only a location into which heat is added or

* Connectors generated as part of models that do not correspond to any existing device are called NULL connectors. GK and HK are calculated internally and are not the responsibility of the user.

subtracted and has no physical significance, the heat rate into the tie is *not* simply the product of the UA coefficient times the temperature difference between the node and the junction. When primed, the saturation temperature is used instead of the junction temperature. *When deprimed, the tie will spontaneously jump to the designated vapor lump and become an HTU tie with the same UA value.* This usually eliminates the error messages that would result when a CAPPMP deprimes with a constant heat rate still being applied. If the CAPPMP reprimes, the tie will reappear at the central junction, and once again become an HTM tie. In either mode, the UA value can be modified. Because this tie can appear and disappear from the vapor side, the user should refrain from specifying the QDOT on that vapor side, although other ties to that lump are permitted. When tied, the CAPPMP macro can be schematically represented as (with path flowrates shown positive for evaporation):



The values of RC, CFC, XVH, and XVL are available to the user in logic blocks for inspection or modification. The ID of the first connector should be used to reference these parameters. In other words, the first (defined upstream) connector may be treated like a CAPIL connector for translation purposes.

The user should understand the assumptions made in this simplified model and their impact on modeling a real capillary evaporator–pump, especially in off–design simulations:

1) The CAPPMP, like any FLUINT element, is fundamentally a one–dimensional component: it must be either deprimed or not. Real devices can exhibit more two–dimensional effects such as preventing vapor backflow and even pumping in some regions of the wick, while simultaneously leaking vapor back into the liquid space in other locations. Furthermore, meniscus recession into the wick is not included.

2) The endpoint lumps are perfectly mixed, having a single thermodynamic state. While NONEQ0/1 routines have been used successfully to model deprime, and the BELACC routine has been used to model the spring–like meniscus, special care is needed to model behavior during deprime, or during clearing of the vapor lump during start–up.

3) While real wicks can exhibit considerable hysteresis (on the order of 50%) between the pressure differences at bubble formation and cessation, only a very small hysteresis (about 4%) is employed internally, purely for numerical stability.

The user should use junctions cautiously at the ends of a CAPPMP model. Junctions are much more likely to cause inadvertent deprime because they must react instantly to changes

Finally, note that a much more rugged but inflexible model of a capillary pump is simply an MFRSET connector and a heater junction (e.g., a junction whose enthalpy is fixed by a call to HTRLMP). The enthalpy of the junction is set to saturated or slightly superheated vapor, and the SMFR of the connector is set to the desired heat rate divided by the enthalpy rise (i.e, that of the junction minus the inlet). Such a model cannot deprime.

4.7.8 Capillary Models and Phase Suction Options

Both capillary models (CAPIL connectors and CAPPMP macros) and phase suction options require additional attention from the user. Because capillary models internally use the phase suction options, these two topics are discussed together.

Twinned tubes and STUBE connectors also manipulate the phase suction options. The primary path is assigned STAT=DLS when two phases are present, and the secondary path is assigned STAT=DVS. While some of the following discussion is therefore relevant to twinned paths, separate sections describe their behavior and usage.

4.7.8.1 Phase Suction Options

Almost any path can be made to extract only liquid or only vapor from an upstream two-phase lump. This is a useful tool for simulating liquid-vapor separators, capillary devices, and stratified tanks. For example, the vapor barrier properties of a filter or small diameter passage may be simulated with a liquid suction option. (Liquid suction options are internally set and reset by the CAPIL and CAPPMP capillary models.) Also, a vent above the liquid surface of a stratified two-phase tank may be simulated with a vapor suction option.

Phase suction options may be set in the FLOW DATA block using the STAT flag, and may be reset during execution using the CHGSUC routine. These options are specified by the following character flags:

LS ..... Extracts liquid from upstream end (liquid suction)
VS ..... Extracts vapor from upstream end (vapor suction)
RLS ..... Extracts liquid from *downstream* end (reverse liquid suction)
RVS ..... Extracts vapor from *downstream* end (reverse vapor suction)
DLS ..... Extracts liquid from both end (double liquid suction)
DVS ..... Extracts vapor from both ends (double vapor suction)

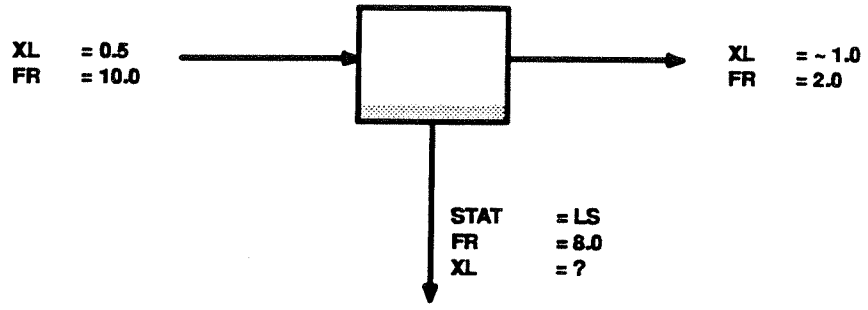
"Upstream" and "downstream" are based on the defined positive flowrate direction. The designation does not change with flowrate reversals. Note that the RLS and RVS options merely save the user from having to reverse a path, use the LS or VS options, and deal with negative flowrates.

Phase suction action can only occur when the upstream lump is in a two-phase state. When the upstream lump is single-phase, the path will resume normal operation (i.e., phase suction action will be ignored). Phase suction action alone cannot prevent all flow through a path if the upstream single-phase lump contains the "wrong" phase.

Note that phase suction options can affect the behavior of a path even if the path is flowing in the opposite direction from the suction action. The path will "see" only the appropriate phase for calculations which depend on downstream properties as well as upstream. Such calculations can include body forces, frictional resistance, and spatial accelerations. Thus, phase suction options may be used on downstream ends of paths merely to affect the properties used.

The quality "seen" by a phase suction path is seldom that of the upstream lump. The PTHTAB routine therefore lists "XL UPSTRM" as the quality used by that path. This effective quality may also be retrieved using the XTRACT subroutine.

The behavior of an upstream lump can be very different depending on whether it is a junction or a tank. (Recall that tanks are treated like junctions in FASTIC.) To understand this difference, assume a liquid-vapor separator has been modeled as a tank with one incoming two-phase path and two exiting paths. One of the exiting paths extracts only liquid when that phase is present:



Suppose that, for whatever reasons, the all-liquid leg tends to operate at a higher flowrate than the vapor leg. If the incoming fluid quality does not exactly correspond to the outgoing phase distribution, then *no steady-state solution exists*. The liquid path will extract all available liquid, driving the quality in the tank to 1.0. At that time, with no available liquid, the liquid path will start to extract vapor (as would a submerged pipe penetration or a deprimed capillary barrier) until liquid is again present. As a result, the quality in the tank and therefore the quality of the vapor exit path hovers near 1.0, while the fluid in the liquid suction path alternates between all vapor and all liquid. A STDSTL run will not converge because there is no time-independent state. (As an aside, note that real fluid systems often lack a time-independent state due to a variety of causes.)

If the separator were modeled with a junction (or were analyzed in FASTIC), the situation would be different. FLUINT will attempt to produce *time-averaged* results if time-independent results are impossible. In the above example, the program will converge in FASTIC with a tank quality of nearly 1.0 (perhaps 0.9999). The liquid path quality will be 0.375 to balance mass and energy. This does not necessarily mean that this path will carry vapor 37.5% of the time, nor that the behavior of the subsystem downstream of that path will represent an average behavior between all-liquid and all-vapor states. However, mass and energy will balance and all phases will be distributed as much as possible in accordance with the phase suction status. Phase suction thus takes on a more liberal meaning. Using liquid suction on a path means that the path will extract *as much liquid as possible* up to a limit of 100% liquid.

The goal of the above methodology is to provide good initial conditions for STDSTL or for transient runs. Because of the lack of a steady-state solution, the results produced by FASTIC *might not* correspond to any physical state. The situation is analogous to orbital average fluxes and temperatures, which are useful as sizing information and as initial conditions for transients, but which will probably never really occur.

The user must be therefore cautioned regarding FASTIC solutions and phase suction from junctions in general, but there are several strong reasons to *not* avoid them. First, *the solution reached by FASTIC is usually a true time-independent state.* Suction path qualities between 0.0 and 1.0 are indications of a time-averaged state. Second and most important, analytic tests with

complex capillary systems have shown that the FLUINT solutions are indeed very close to the average behavior of the oscillating system, and are much easier to study than the true oscillating solution.

As a result of active phase suction options, the quality of a junction or FASTIC tank is not necessarily equal to a quality resulting from a perfect mixing of incoming flows. If this were true, the quality of the fluid inside the aforementioned liquid-vapor separator would be 0.5. Actually, the time-averaged quality will be nearly 1.0 because of the tendency of the liquid path to extract all available liquid. This shifting is taken into account by the code, and has important ramifications on liquid inventory and heat transfer calculations. Suppose the tank in the above example were tied to a cooler wall node, and that the UA value is calculated for film condensation. The dryer the tank, the higher the UA coefficient and the heat rate out of the tank will become, causing more condensation and lower quality. These opposing forces will reach an equilibrium quality lower than that of an adiabatic tank, and/or will allow more liquid to flow out of the liquid suction path. Such considerations are taken into account by the code, and should be remembered by the user when manipulating UA values for HTU or HTUS ties.

Constant enthalpy in a two-phase state implies nearly constant quality. When phase suction options are active out of a two-phase FASTIC tank or junction whose enthalpy has been frozen by a call to HTRLMP, outflowing suction paths will have qualities of 1.0 or 0.0 since the desired phase is assumed to be available. The lump QDOT will be calculated to balance total energy flow through the heater lump. This is consistent with the quality shifting as mentioned above; using phase suction options, *phases may appear to move from inlet to outlet without ever mixing*. As with any lump, if the quality in the heater lump is either 0.0 or 1.0, all exit paths will carry that quality.

4.7.8.2 Capillary Primed vs. Deprimed Decision

Because of the need to manage liquid position in low gravity, capillary devices are common in spacecraft systems. In FLUINT, the available capillary models include CAPIL connectors and CAPPMP macros. These models require special attention because (1) they internally use the phase suction options, and are therefore subject to the previous discussion, and (2) they can exist in one of two very different states: *primed* or *deprimed*. FLUINT's definitions of these terms, detailed in the next paragraph, are not necessarily intuitive, having more to do with whether or not a steady meniscus exists than whether the wick is dry or wet.

A capillary device such as a wick, groove, slot, or tubule is primed if it is full of liquid and is adjacent to vapor whose pressure is greater than or equal to the pressure of the liquid but less than or equal to that supported by the capillary radius of the device. (The capillary pressure limit is equal to $2 \cdot \sigma / RC$, where σ is the fluid surface tension and RC is the effective two-dimensional capillary radius of the liquid meniscus in the capillary structure.) Otherwise, it is deprimed. For the purposes of the following discussion, "deprimed" means that the flowrate becomes a function of the resistance and the current pressure gradient. (For the advanced user, the distinction between primed and deprimed may be summarized concisely: the connector GK is zero if primed, and positive if deprimed.) If heat is added to a primed device, it may be able to evaporate the liquid into the higher pressure vapor space, thereby becoming an *evaporator-pump*. On the other hand, *a flooded capillary device is considered deprimed in FLUINT even if it can still block vapor flow* (e.g., heat inputs are low and the pressure gradient is less than the capillary limit).

Unfortunately, capillary devices can be easily deprimed (more so analytically than in practice). First, if the pressure differential exceeds the capillary limit, any fluid in the higher pressure end will be able to flow back into the lower pressure side. This dries out the capillary structure, which can then only be rewetted if the flow is reversed or if sufficient condensation occurs. Second, if the liquid pressure is greater than that of the vapor, liquid will blow through into the vapor. Third, if too much liquid appears in the "vapor" end, it causes the device to deprime and the liquid will flow back to the lower pressure end until it is exhausted, at which time the device might reprime. Fourth, a heated device (i.e., a CAPPMP) cannot stay primed if its own internal resistance causes an excessive pressure drop from the liquid lump to the capillary interface. This *induced pressure drop* occurs within the capillary structure itself, and is a function of the heating rate and therefore the mass flowrate. Once deprimed, such devices usually do not reprime on their own. Figure 4–12 depicts the process used by FLUINT to determine whether or not a device is currently primed.

In a system with N capillary devices, there can be up to 2^N possible and valid steady-state solutions corresponding to whether or not each device is deprimed. Although *valid,* the solutions with deprimed devices are usually not *desired.* If the totally primed case were possible and all other considerations equal, the probability of arriving at that case would only be about $100/2^N$ percent. Unfortunately, other considerations are not equal. Because of the analytic fragility of the primed state and the typically rough nature of steady-state calculations, FLUINT would almost always find the case where most if not all of the capillary devices are deprimed.

Instead of posing the question: Can these devices stay primed under these conditions?, the user must instead *assume that they do stay primed* and then see if a valid solution is possible given that assumption. This is the purpose of the SPRIME ("stay primed") routine described in Section 6. SPRIME can be called in FLOGIC 0 during steady-states to force a CAPIL or CAPPMP model to stay primed. SPRIME will make the minimum adjustments necessary in the states of the two endpoint lumps, and will report the changes necessary to keep the device primed. Adjustments made at the start of a steady-state run can be ignored. If no adjustments are made at the end of the run and the model converges, then the device *can* stay primed under the current conditions. If the model does not converge after a reasonable number of iterations *and SPRIME keeps repeating the same adjustments or makes increasingly greater adjustments,* then the device cannot stay primed. The repeated adjustments made by SPRIME should provide a clue as to why the device didn't stay primed, and a new run without that particular SPRIME call should reveal the true deprimed steady-state solution.

The user should also take precautions to assure that capillary devices do not artificially deprime during a transient run. SPRIME can also be used for this purpose, but other methods may be more appropriate.

First, the capillary radius (RC) may be nonpositive, which signals the assumption of an ideal capillary structure that can provide or sustain an infinite pressure head. This usage can only prevent certain causes of deprime.

Second, the adjacent tanks may be given a compliance factor to take into account the fact that the liquid control volume is far from rigid because of its proximity to a vapor control volume. The assumption of a rigid container can easily lead to pressure surges that can deprime the device. A small but nonzero COMP factor eliminates such surges, and produces a more realistic response. *THIS SHOULD BE CONSIDERED MANDATORY FOR ALL CAPILLARY DEVICES.* See Section 4.7.11 for methods of computing COMP factors. Also, the BELACC simula-

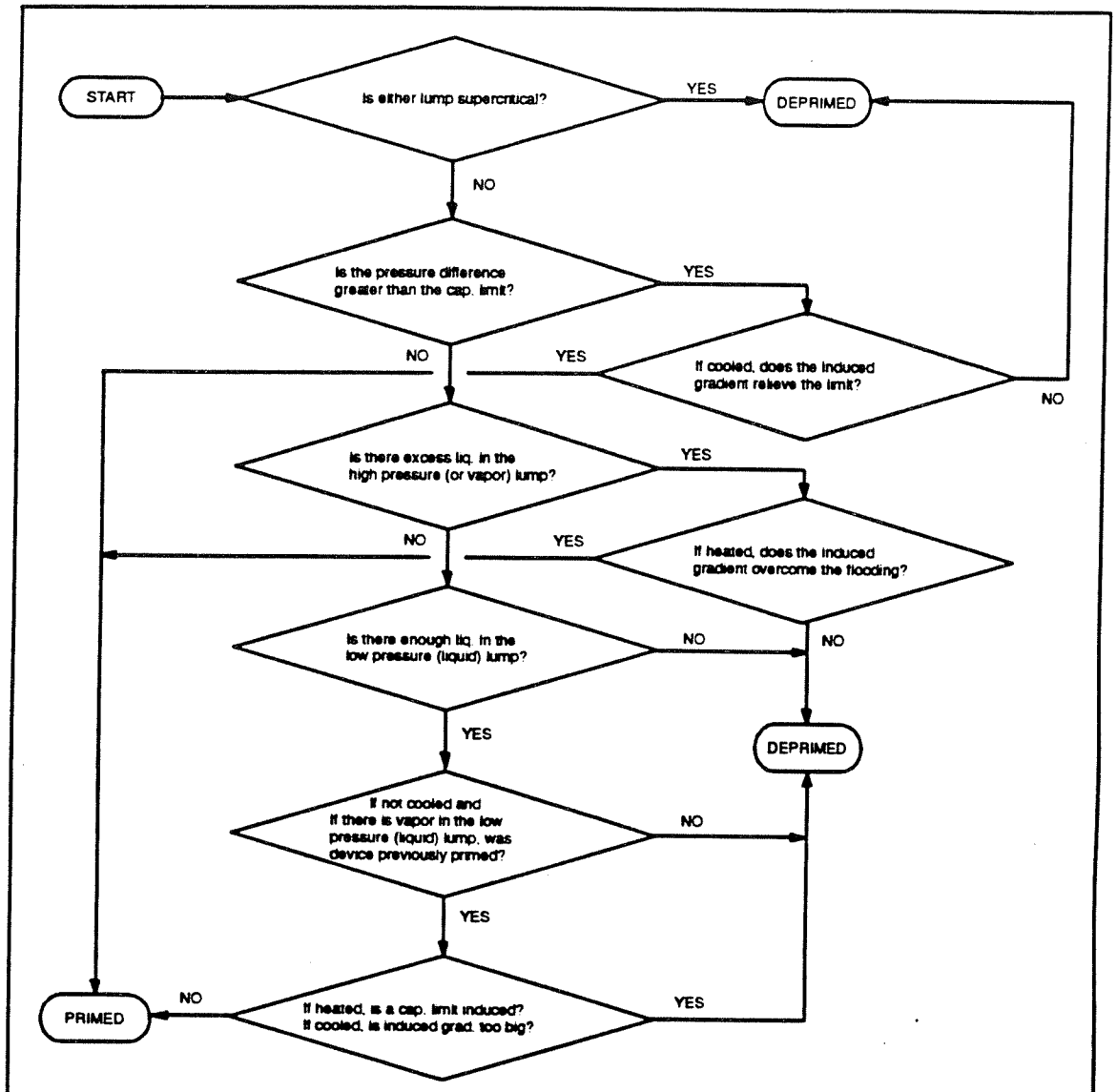


Figure 4–12 Algorithm for Determining Prime/Deprime Status

tion routine has been successfully used to include the spring–like effects of the meniscus if two tanks are used as endpoint lumps, representing the vapor and liquid spaces. This provides a realistic basis for the liquid tank compliance as well as conserves total evaporator volume.

Third, the primed/deprimed decision can be affected for CAPIL connectors and CAPPMP macros by helping the program decide: how dry is dry? The remainder of this section discusses this feature.

FLUINT assumes perfect mixing and therefore even distribution of liquid within a two-phase lump. To be consistent with this assumption, any liquid in the higher pressure side (or designated vapor side) of a capillary device could deprime the device by being absorbed back into the lower pressure side (or designated liquid side). If heat is being added to a CAPPMP when it deprimes, the flowrate will drop and the junction might overheat.

Most real capillary devices may be fully primed and operational at outlet qualities less than 1.0; the liquid in the outlet is either boiled off or swept out of the outlet. Remember that a quality of 0.5 can easily correspond to a void fraction greater than 0.99; the liquid would hardly be visible and may not be a factor in the primed/deprimed decision.

Two factors are available to assist in the dryness decision in CAPIL connectors and CAPPMP macros: XVH and XVL. These factors are simply qualities above which a lump can be considered to be mostly vapor, at least in terms of a primed vs. deprimed decision. (These factors are also applied to the liquid side (lower pressure lump) to determine whether or not it is wet enough.) A lump with quality greater than XVH is allowed to fill with liquid (for whatever reason) until the quality reaches XVL, which may cause deprime. A lump whose quality is less than XVL must reach XVH before a reprime can occur. In other words, XVH should be greater than XVL to provide some stabilizing hysteresis, but they may be equal. The default values are XVH=0.99 and XVL=0.95.

For example, if XVH=1.0 and XVL=0.99, then the higher pressure lump would have to be initially dry (XL=1.0) to permit priming, but may drift as low as XL=0.99 before causing a deprime. Once the quality is below 0.99, the lump would have to be completely dry to allow a reprime to occur. As a rule of thumb, the user should decide which quality corresponds to "mostly dry", and then set XVH and XVL to bracket that value by a few hundredths. If a quality of 0.8 is chosen, then XVH=0.82 and XVL=0.78 would suffice, while XVH=0.85 and XVL=0.75 would have more hysteresis if a prime/deprime oscillation is encountered. Setting both XVH and XVL to 0.8 is legal but may be susceptible to oscillations. A lower value of XVH and XVL means the device is more likely to become or stay primed. However, too low a value can produce misleading results. Recall that the dryness factor apply to the lower pressure (liquid) end as well. *A lump with a quality above XVH is considered "dry" no matter which side of the device it is on.* This means that the program may decide upon a deprimed solution on the basis of insufficient liquid. Values less than 0.5 should be used with caution, and values less than 0.1 are not recommended.

As noted above, a low value of XVH and XVL means the device is more likely to become or stay primed. When used in conjunction with RC less than or equal to zero (i.e., infinite capillary head), a low value of XVH and XVL (say 0.5) is usually a superior alternative to an SPRIME call. In both cases, the user must assume a primed solution in FASTIC or STDSTL and then verify that such a solution is possible. Of course, both methods may be used together.

SPRIME may perturb a solution unnecessarily, whereas the use of zero RC and small dryness factors merely eliminates or avoids most deprime mechanisms. As noted above, continuous SPRIME adjustments are a sign of lack of a valid primed solution. Inability to prime when RC, XVH, and XVL are returned to realistic values after convergence is similarly an indication of no valid primed solution.

There are other uses for the dryness factors XVH and XVL. To illustrate, recall that FLUINT assumes one-dimensional paths. A real evaporator-pump may be partially primed, pumping in some locations, and deprimed in others. To allow for pumping while at the same time allowing liquid to flow back into the lower pressure inlet, a CAPPMP could be placed in

parallel with a CAPIL connector. The XVH and XVL values of the CAPPMP would be low, signaling a tolerance to liquid in the outlet, while the XVH and XVL values of the parallel CAPIL connector would be high (say 1.0 and 0.99), allowing the device to deprime and pass liquid back to the inlet. The situation is analogous to countercurrent flow modeling in a vertical two-phase line, where again the solution is to use parallel (twinned) paths to allow liquid to fall and vapor to rise.

The above discussion touches on the occasional inadequacy of the one-dimensional and perfect mixing assumptions when applied in combination with capillary modeling. As an expansion of this topic, the user should note the difficulties that can arise when using tanks on either end of a capillary device. Consider one tank containing subcooled liquid, and another adjacent to it containing superheated vapor at a slightly higher pressure. These two tanks are separated by a CAPIL or CAPPMP. Slight pressure perturbations might raise the liquid tank pressure higher than the vapor tank pressure. Due to the perfect mixing assumption, addition of a little cold liquid into the vapor tank would cause that tank to *cold shock* (e.g., collapse to liquid). Given the same scenario, if the pressure differential exceeds the capillary limit, warm vapor would begin to flow into the liquid side. Again because of perfect mixing, the liquid tank pressure would rise instead of displacing liquid, and that rise would reprime the device, causing a perpetual oscillation between primed and deprimed states.

In either case, the NONEQ0 and NONEQ1 routines (see Section 6) can help, but the user must "kick-start" their nonequilibrium operation, noting that they only begin to deviate from perfect mixing when the void fractions are between 0.01 and 0.99 by default. Although the user may reset these limits using the NONEQS option, by the time the above tanks have achieved a reasonable void fraction, the trouble has already occurred.

4.7.9 Symmetry and Duplication Options

Duplication options allow the user to identify parts of the fluid submodel that are identical and analytically repetitive. In doing so, considerable computational savings can be realized. In most real fluid systems, there are numerous examples of parallel sections that can be considered identical for analytic purposes, but whose heat transfer and pressure drop characteristics cannot be properly modeled with a single passage having larger "effective" dimensions. Examples include tube bundles or finned sections in a heat exchanger, capillary evaporator segments in a coldplate, and grooves in a heat pipe. Identical sections also can occur as a result of redundancy or symmetry in the system itself, such as parallel subsystems. FLUINT has a general mechanism for identifying such repetitions and significantly reducing both the input effort and the run times. In fact, it is strongly recommended all sections that can be *assumed* to be identical should be modeled as such to simplify the input and to speed the execution. Otherwise, the focus of the analysis may inadvertently shift toward tracking interactions between parallel passages.

Duplication options are designed to be very general to take advantage of types of symmetry and redundancy that cannot be foreseen by the authors of the program. Thus, while these options may appear to be simple, the user will find it well worth his time to study them thoroughly to both maximize his usage of them and to prevent errors, which can be very difficult to detect. Duplication options are also common sources of many advanced modeling tricks.

Duplication options may be applied to paths and/or ties, as described below.

4.7.9.1 How Path Duplication Factors Work

Identical subnetworks and other model symmetries are specified using *duplication factors* that can be attributed to any path. If there are ten identical paths between two lumps, a single path is described and given a duplication factor of ten. Internally, FLUINT stores and analyzes only one path, but the effect of this path on the two end lumps is magnified tenfold, as illustrated in Figure 4–13. For example, if there were a bundle of 1000 microtubules between two control volumes, the user would describe two tanks and one path (perhaps a tube or a STUBE or CAPIL connector) with a duplication factor of 1000. The default value for all duplication factors is 1.0.

Actually, there are *two* duplication factors for every path—one for the *defined* upstream end and one for the *defined* downstream end (called DUPI and DUPJ, respectively). This allows considerable flexibility in describing unusual types of symmetry and redundancy, but using two factors instead of one may seem awkward at first. The key to understanding duplication factors is this: *the defined upstream lump "sees" a path DUPI times, while the defined downstream lump "sees" the same path DUPJ times;* DUPI and DUPJ need not be equal.

In order to comply with the configuration shown in Figure 4–13, *both* DUPI *and* DUPJ would be specified as 10.0 (input options are designed to set both factors using a single keyword 'DUP'). In many fluid systems, however, the identical sections cannot be modeled using a single duplicated path. Such a case is illustrated in Figure 4–14, where the identical portions can be an arbitrarily complex subnetwork *and can even have duplicated subsections within that subnetwork.* This example demonstrates the utility of using two duplication factors per path.

There are no restrictions on the level of "nested" duplication, and very few restrictions on the value of any duplication factor. In fact, a subnetwork (or a single path, which is simply a special case of the term *subnetwork*) does not even have to flow back into the system at all, as

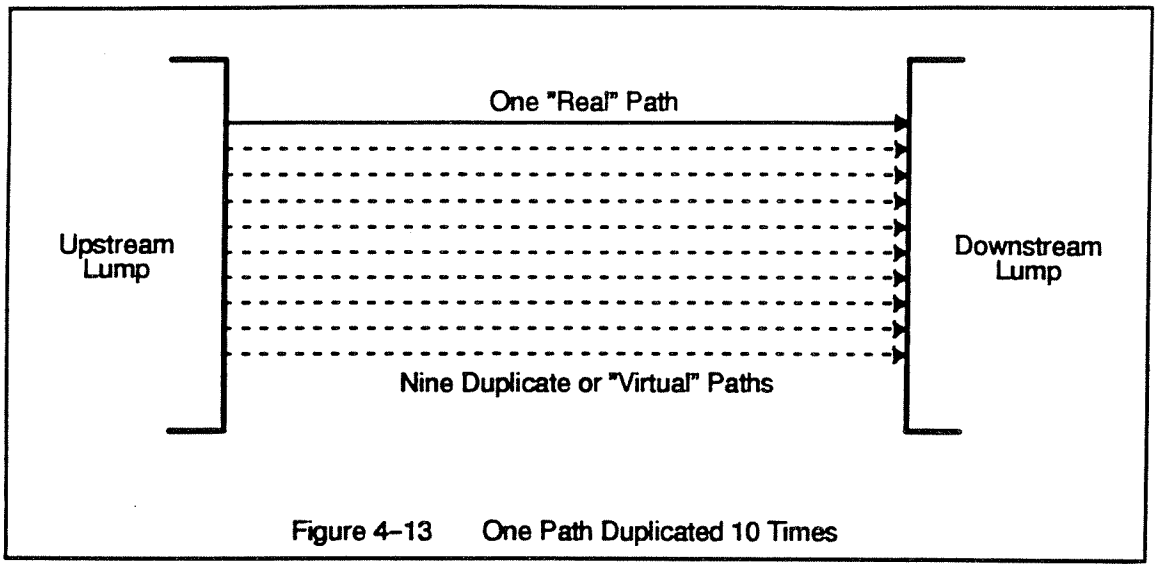


Figure 4–13 One Path Duplicated 10 Times

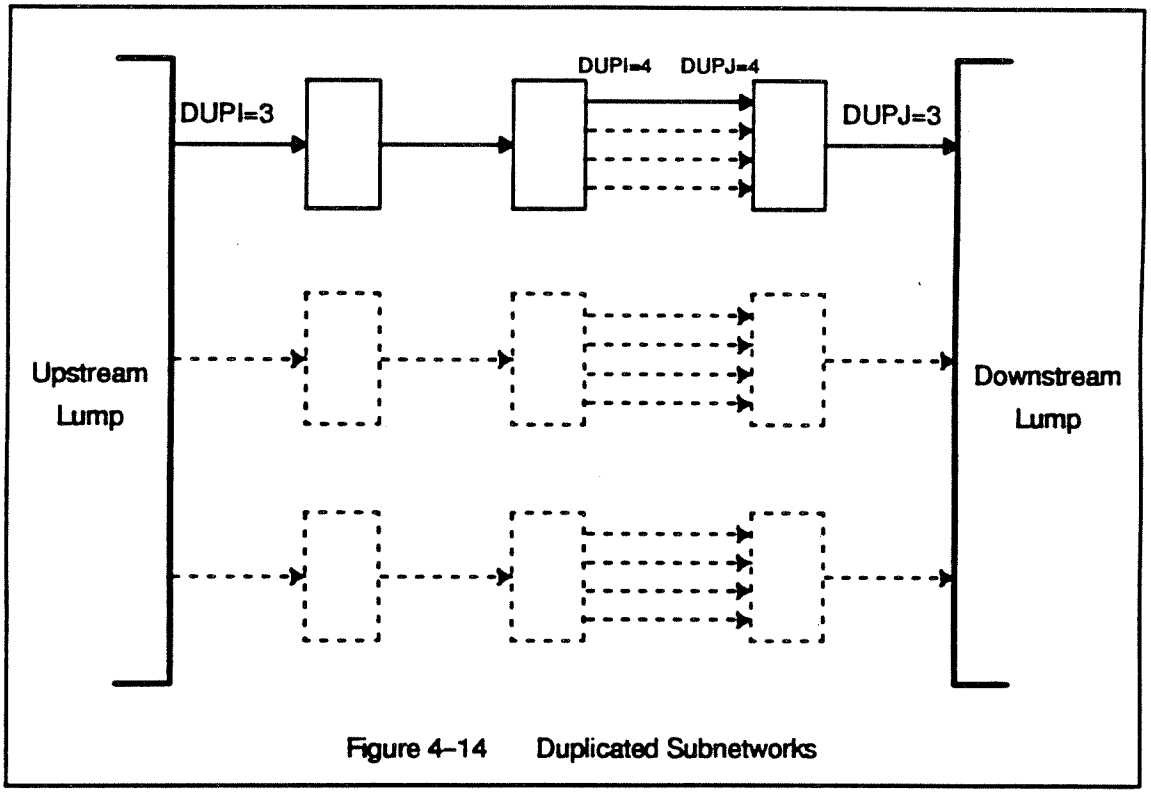


Figure 4–14 Duplicated Subnetworks

shown in Figure 4–15. In such a system, the duplicated or *virtual* subnetworks exist only from the point of view of the upstream lump. Although the examples in this section cover most cases of symmetry in fluid systems, the user is welcome to experiment with more complex levels of duplication. Note that the FLOMAP and PTHTAB routines print the current values of duplication factors along with other path information to help debug such complicated symmetry.

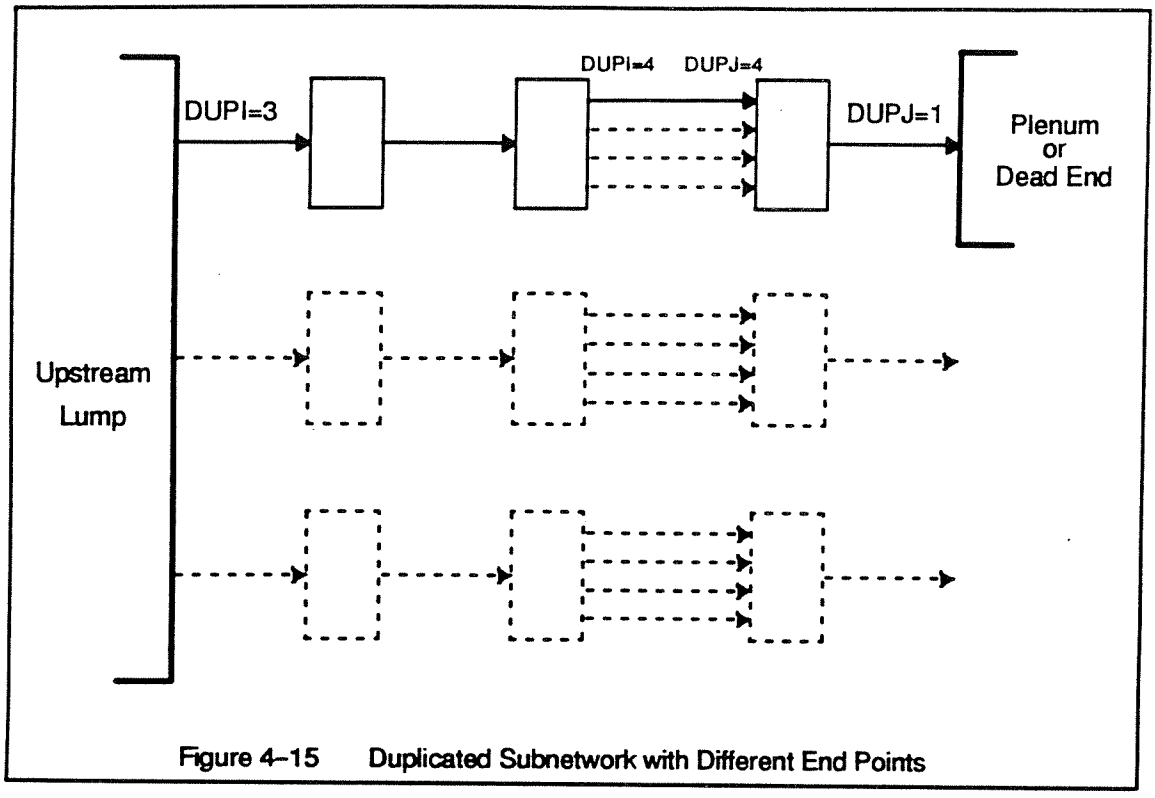


Figure 4–15 Duplicated Subnetwork with Different End Points

The user must take care to assure that if a duplicated section flows back into the main network, the total upstream and downstream duplication factors balance each other. Otherwise the virtual subnetworks will act as unintentional sources or sinks of mass and energy, which may result in illegally high or low pressures.

For discussion purposes, the upstream and downstream lumps that start and end a duplicated section are called *manifolds*. This name is meant to stress an underlying assumption—virtual subnetworks share the same boundary conditions since they all flow out of and/or into the same lump. Thus, the FLUINT model shown in Figure 4–16a cannot make use of duplication factors unless all upstream lumps can be combined into a single lump, and all downstream lumps can be similarly modeled as one lump. This assumption (Figure 4–16b) is usually equivalent to assuming a negligible pressure gradient in the manifold, which is an excellent assumption since that is the purpose of using a manifold in a real system. The user is strongly advised to make such assumptions if possible because of the tremendous savings in time for both the computer and the user.

With respect to twinned paths, note that only each twin shares the same definitions of DUPI and DUPJ, just as they share only one definition of DH, TLEN, etc. Refer only to the duplication factors of the primary twin when making changes in logic blocks.

In order to demonstrate some of the concepts discussed here, some FLOW DATA input examples are presented here. Refer to Section 3.11 for specific input formats. Note the use of 'DUP' to set both 'DUPI' and 'DUPJ', although no such short cut exists in logic block references. The default values are always 1.0.

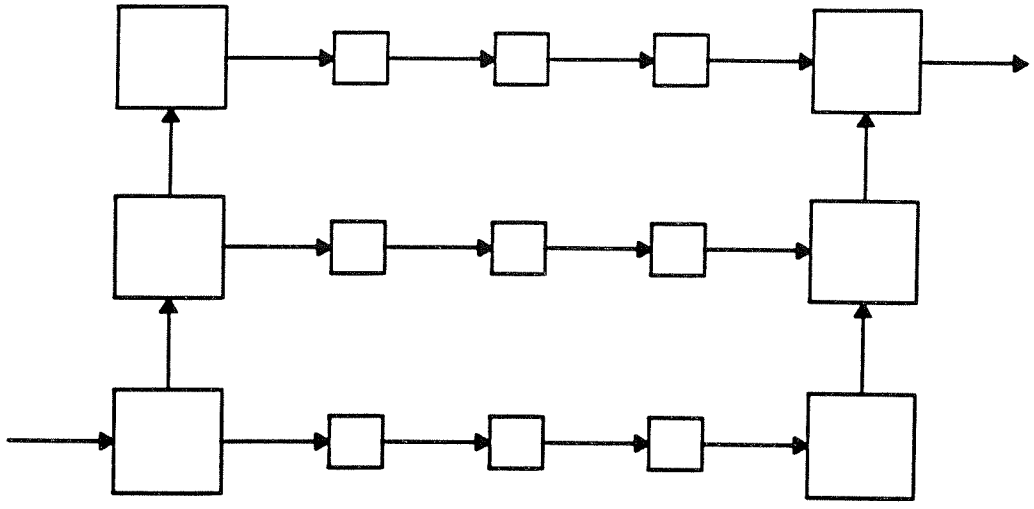


Figure 4–16a Subnetworks That Cannot Use Duplication Factors

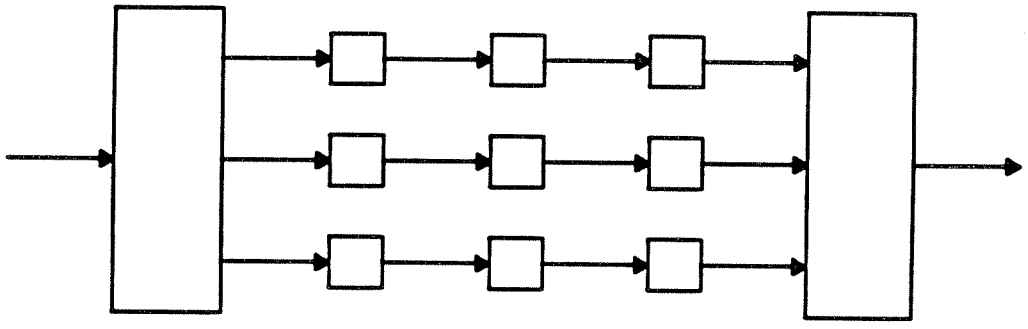


Figure 4–16b Above Model After Duplication-Enabling Assumption

EXAMPLE 1—In the following example, two paths are generated between lumps 10 and 20. The first is a tube that is duplicated 300 times between these lumps. The second is an STUBE connector that appears once from the point of view of lump 10 but twice from the point of view of lump 20:

```
PA TUBE,100,10,20,      TLEN    = 1.0       $ EXAMPLE 1a
     DUP        = 300.0 ,       FR      = 22.0
C
PA CONN,200,10,20,      DUPJ    = 2.0       $ EXAMPLE 1b
     DEV        = STUBE,        DH      = 0.5/12.0
```

EXAMPLE 2—In this example a center-discretized LINE with 10 segments is duplicated 300 times between manifold lumps 100 and 200. In a second example, an upstream-discretized LINE command is used along with additional subblocks necessary to accomplish exactly the same result as the first example:

```
M LINE,1,C,1,1,100,200, NSEG=10            $ EXAMPLE 2a
     TLENT      = 10.0,          DHS      = 0.75/12.0
     DUP        = 300.0
     PA         = STUBE,         LU       = JUNC
C
M LINE,1,U,1,2,10, NSEG=9                  $ EXAMPLE 2b
     TLENT      = 9.0,           DHS      = 0.75/12.0
     PA         = STUBE,         LU       = JUNC
PA DEF,         DH       = 0.75/12.0
                TLEN     = 0.5
PA CONN,1,100,1,   DUPI=300.0,DEV=STUBE
PA CONN,11,10,200,DUPJ=300.0,DEV=STUBE
LU JUNC,10
```

4.7.9.2 How Tie Duplication Factors Work

Heat transfer ties also use duplication factors. Nearly all of the foregoing discussion applies to tie duplication factors directly or by analogy, and hence will not be repeated here except to clarify concepts.

Tie duplication factors are DUPL and DUPN, which also default to 1.0. *The lump "sees" a tie DUPL times, while the node "sees" the same tie DUPN times;* DUPL and DUPN do not have to be the same value. To avoid confusion in macrocommands, there is no input option for ties analogous to the path "DUP" keyword. Mapping routines print duplication factors and their effects, but unfortunately the TIETAB routine does not print these factors due to page width limitations.

One of the principal uses of tie duplication factors is to contain virtual networks created by DUPI and DUPJ within the same fluid submodel, without affecting the thermal submodel. For example, if a heat exchanger has 10 identical parallel passages, then path duplication factors of 10.0 would apply at the beginning and end of the passages. However, if the corresponding thermal nodes modeled the whole heat exchanger (instead of one tenth of it, i.e., one identical passage), then the DUPN factors could be set to 10.0 such that the thermal model "saw" the effects of all ten fluid sections. Other methods of dealing with symmetry in thermal models are presented below.

As another example of tie duplication factors, suppose a detailed model of a pipe and fin were generated, and the user wishes to exploit the fact that the centerline of the pipe correspond to the plane of symmetry—the fins on either side are identical, and therefore only half of a pipe

cross-section is needed for thermal boundary conditions. In this case, the fluid submodel should correspond to the entire pipe for realistic pressure drop calculations, but each tie to the thermal "half wall" can use a DUPL factor of 2.0 such that each lump in the pipe model "sees" two identical nodes (one on each side).

One slight difference exists between the way path duplication and tie duplication factors are implemented. Tie calculations are suspended in the limiting case when both DUPL = DUPN = 0.0, which is taken to mean that tie has been turned off. This special case exists to prevent property range limit warnings when the wall temperature is temporarily out of the valid range (e.g., for freeze analyses) and the user wishes to temporarily disable the tie.

4.7.9.3 Referencing Duplication Factors in Logic Blocks

DUPI, DUPJ, DUPL, and DUPN may be used as variables within logic blocks in the same manner as FR, UA and other path or tie descriptors (e.g., [fsmn.]DUPIn). The user should *not* reset these values in FLOGIC 1. Also, note that "DUP" is only used as an input aid and does not correspond to any path descriptor—in logic blocks, both DUPI and DUPJ must be reset if duplication factors for both ends of the path are to be changed. For twinned paths, refer only to the duplication factors of the primary twin since there is only one definition for each pair.

4.7.9.4 Important Impacts of Duplication Factors

HTN, HTNC, and HTNS Ties—A path that is duplicated with respect to a neighboring lump is duplicated in every sense—including heat transfer characteristics. When a HTN, HTNC, or HTNS tie is made between a node and a lump, one (HTN, HTNS) or two (HTNC) paths are named to convey the characteristic dimensions, including heat transfer area. Thus, a duplication factor of 10.0 on the upstream end of a tube or STUBE connector (or set of twins) will increase the heat transfer area attributed to the upstream lump by a factor of 10.0, and hence will increase the heat rate of that lump by the same factor. This is very useful in characterizing heat exchangers with many small passages such as those created by fins: an increased heat transfer area is available while the effects of the small diameter on the flowrate and the heat transfer coefficient are preserved.

There is one subtlety regarding virtual paths involved in HTN and HTNC ties. There are currently no restrictions on the location of a path involved in such a tie—unlike HTNS ties, it does not have to be adjacent to the lump through which heat is transferred. Because of this freedom the user should remember that *magnification of the heat rate of a tie will only occur if the lump and path involved in the tie are adjacent.*

For HTNS ties, another potential problem exists if flowrates reverse and unequal (asymmetric) path duplication factors are used (i.e., DUPI $\neq$ DUPJ). Since these duplication factors affect the amount of heat transfer area 'seen' by the lump, the value of UA may change by the ratio DUPI/DUPJ for HTNS ties if flows reverse and the "STAY" command has not been issued. A warning is produced at the start of the processor if it detects the potential for such a condition given the *initial* values of DUPI and DUPJ and the absence of the "STAY" command, but no further actions or checks are performed thereafter.

Thermal Submodels—There are no corresponding duplication factors in thermal submodels. As described below, *imbalances in the energy flow of a thermal submodel will result if it is tied to a duplicated fluid subnetwork but is not isolated or tie duplication factors are not used.* (Imbalances are allowed and in fact are common in fluid submodels, but can slow steady-state convergence and can cause confusion.) As an alternative to duplication factors, dual one-way conductors (either linear or radiation) can be used to emulate thermal duplication factors. Thus, the thermal model of a fluid container can be made to mimic the way that redundancies and symmetries are exploited in the fluid model.

To demonstrate this technique, Figure 4–17 shows how to make one node appear duplicated three times with respect to a second node, while the second node would seem to appear only once with respect to the first node. This is analogous to the DUPI=3.0 part of the major subnetwork in Figure 4–14, and could be used to model the pipe wall across this "duplication discontinuity."

In fact, the SINDA/FLUINT thermal user may find such modeling techniques can significantly reduce the size of purely thermal models. The thermal user should also pay attention to the next subsection dealing with modeling tricks, since many of these tricks also apply by analogy to thermal-only models using dual one-way conductors. Refer to Section 3.10 for a discussion of one-way conductors, and refer to Sample Problems E and G for a demonstration of such modeling practices.

Energy Balances—The use of dual one-way conductors and virtual, diabatic fluid subnetworks may affect the overall steady–state energy balance check: the submodel can be considered converged even though the system-level energy flows do not balance. The imbalance is created by the fact that duplicated sections can appear to the full network as sources or sinks of mass and energy. (In mathematical parlance, the coefficient matrix is asymmetric.) This fact is recognized in both thermal and fluid energy balance checks, and may result in a message such as "ENERGY STABLE BUT NOT BALANCED."

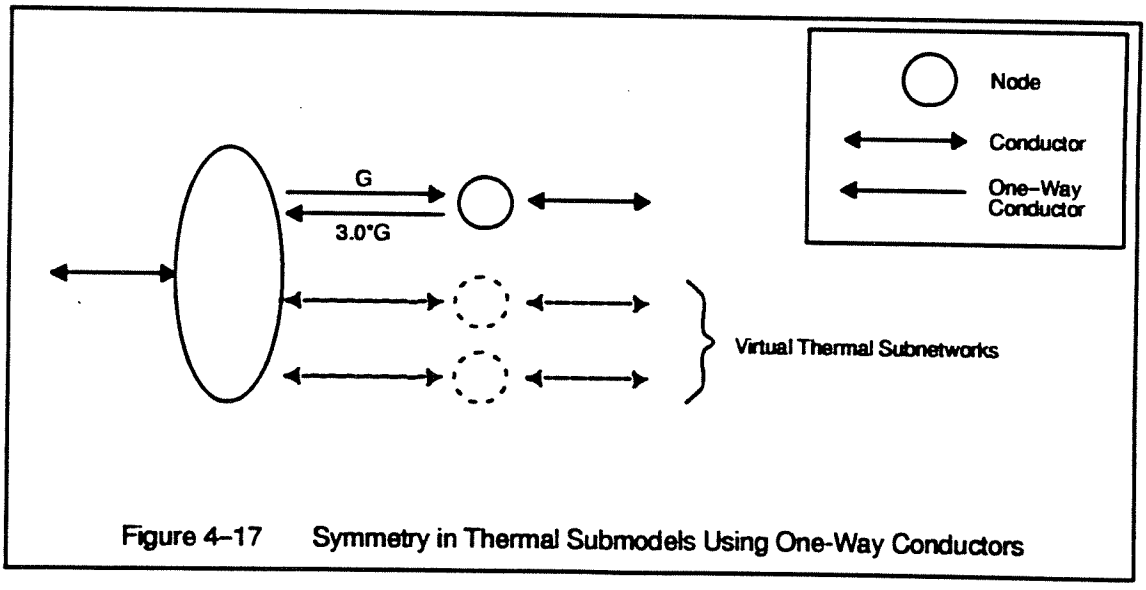


Figure 4–17 Symmetry in Thermal Submodels Using One-Way Conductors

This message should not be construed as a warning that the results are erroneous or inaccurate. Rather, it is intended to warn the user to make sure the final answers are consistent with the use of the duplication factors and/or one–way conductors. However, in thermal submodels another cause of this message exists which *does* indicate inaccuracy: large conductors and small temperature differences can cause steady state solutions to stall, and this lack of progress may be mistaken for asymmetric networks. If such is the case, the user should either reduce the convergence criteria and continue, or correct the large conductors if they are in error, or eliminate the large conductors by combining adjacent nodes.

When in doubt, consider using the lump–by–lump energy checks (negative REBALF) and node–by–node energy checks (EBALNA). While they are usually less preferable than system–level checks, these element–level checks are not sensitive to one–way conductors or duplication factors.

4.7.9.5 Modeling Tricks Using Zero and Noninteger Duplication Factors

In most models, duplication factors will be integral (although they are REAL in Fortran). This section describes the implications of using zero and noninteger duplication factors. Negative factors should be avoided.

Zero Duplication Factors—A zero duplication factor may be used to make a path or tie "disappear" from the point of view of an adjacent lump or node. From the point of view of a path with zero duplication factor, the relevant lump will still appear, but changes in the path will not affect the lump. Note that this does not reduce execution time since *the "zeroed" subnetwork does not itself disappear, merely its effects on the rest of the network.* Zero path factors may be used to simulate a valve closing next to the lump without causing undue pressure/flowrate changes in the zeroed subnetwork (through which flow continues), or to dynamically add, delete, or exchange subnetworks. Zero tie factors may be used to shut off heat transfer through that tie,* perhaps to turn on another parallel or nearby tie. Using 0.0 on one end of a path or tie and 1.0 on the other end is analogous to a one-way conductor in thermal models. Clearly, zero duplication factors are a powerful tool and will certainly provide a mechanism for future modeling tricks.

Note that tanks and junctions must have at least one active paths adjacent to them. Also, zeroing all ties to a lump will zero its QDOT—the user does not regain control of the QDOT, as he does if the tie is moved using the PUTTIE routine.

Noninteger Duplication Factors—Fractional factors (such as 0.1 and 2.5) can also be useful in certain modeling instances. To a lump, a fractional path has the normal diameter but carries DUP times the flowrate and has DUP times the heat transfer area (where DUP is either DUPI or DUPJ). One possible use of such factors is to divide a fluid component in half or some other fraction. For example, five parallel pipes between two 1.0 ft^3 manifolds could be conceivably be modeled as 2.5 parallel pipes between two 0.5 ft^3 manifolds.

* As noted before, if both DUPL and DUPN are zero, tie calculations are suspended. The same is not true for path duplication factors; path calculations are always performed.

Fractional duplication factors may be used to replace integral factors. For example, a path with DUPI=6.0 and DUPJ=1.0 may be replaced with a path that has DUPI=1.0 and DUPJ=1.0/6.0. The only difference would be that the first path carries one sixth of the flowrate of the second path (i.e., appears six times rather than once). This trick is useful when upstream- or downstream-discretized LINE or HX macros (opt=U or opt=D) are duplicated. In those cases, the duct starts or ends with a lump that appears duplicated with respect to the rest of the network. A fractional value on a path adjacent to such lumps can be used to combine or split the flow appropriately without introducing an unnecessary path.

Fractional tie duplication factors are useful because they allow dynamic variations to HTN and HTNC ties—they can be used as multiplication factors to augment or degrade heat transfer calculations, perhaps to add fouling or to parametrically investigate sensitivity to heat transfer coefficients. Because area fractions for these ties cannot be changed during the processor execution (i.e., can only be set in FLOW DATA), tie duplication factors provide an alternative for dynamically varying the heat transfer area on any tie.

4.7.10 Gravity and Acceleration Body Forces

The user may impose body forces due to gravity or accelerations on all active fluid submodels. This option is exercised by attributing an elevation to any lump in the network and an acceleration (such as "g") in that direction. More generally, the location of any lump may be given in three Cartesian coordinates, and the acceleration may be described as three components of a vector. While one dimension is sufficient for ground-based systems, three dimensions are available to allow simulations of launch and orbital maneuvering using the same model.

Flow regime mapping (IPDC=6) and slip flow modeling with twin paths are both affected by the direction and magnitude of body force terms, as described in later sections.

The LMXTAB output routine is useful for tabulating lump coordinate locations and listing the current components of the acceleration vector.

4.7.10.1 Definitions

The body force can be thought of as a superimposed pressure drop or gain on each path, similar to the HC factor that the user may apply to tubes and STUBE connectors. This additional pressure force is a function of the effective path density and the dot product of the acceleration vector and the *path vector*, where the path vector is the vector pointing from the defined upstream lump to the defined downstream lump:

$$\Delta P_a = \varrho_{eff} \left(a_x (x_i - x_j) + a_y (y_i - y_j) + a_z (z_i - z_j) \right) F$$

where:

ΔP_a ..... Superimposed acceleration pressure force, i to j direction
ϱ_{eff} ..... Effective path density
a_x, a_y, a_z ..... Acceleration vector components
x_i, y_i, z_i ..... Coordinates of defined upstream lump
x_j, y_j, z_j ..... Coordinates of defined downstream lump
F ..... Unit conversion factor for English units

The coordinate locations of each lump are input by the user as CX, CY, and CZ. These variables have units of length (ft or m), and are defaulted to zero—no elevation differences between lumps. Note that these coordinate locations are used for body force calculations only. There is no relation between lump coordinates and tube or STUBE connector lengths. For flexible modeling, paths may be longer or shorter than the distance indicated by endpoint coordinate locations. Furthermore, as evidenced by the default values, any number of lumps may occupy the same location in three-space.

A lump is defined in this manual as having a single thermodynamic state. Actually, that definition should be generalized to include *a single coordinate location*. Thus, to simulate the body force felt by a piping penetration submerged in a vertically stratified tank, the coordinate location of that tank might be input as that of its surface, with the elevation of the penetration point being specified according to its depth. If the surface level changes, the user must update

the coordinate location accordingly. Similarly, specify more than one distinct elevation or coordinate position will require more than one lump, even if thermodynamic state (e.g., temperature and pressure) is the same. Sample Problem B uses three separate plena with identical states for just such a purpose.

The components of the acceleration vector, ACCELX, ACCELY, and ACCELZ, are *global* control constants like ABSZRO and therefore apply to all submodels. To facilitate modeling, *the ACCEL vector is taken to be in the opposite direction of the force on the fluid,* consistent with the above equation. Therefore, if the CZ array represented the *elevation* of lumps from some arbitrary reference point, then use ACCELZ = +g to simulate gravity. For body forces due to acceleration, remember that an acceleration in one direction yields a net force on the fluid in the opposite direction; the ACCELX/Y/Z vector should be input as the direction in which the fluid container is accelerating.

Acceleration vector components have units of either m/s^2 or ft/hr^2. To be consistent with the conventions dictated in Appendix D, the English units for acceleration are ft/hr^2 and *not* the more familiar ft/s^2. In these units, the acceleration of gravity is equal to $32.174*3600^2 = 4.17E8$ ft/hr^2. In SI units, the acceleration of gravity is 9.81 m/s^2.

Recall that no thermodynamic state can be directly attributed to a path. The effective density of each path is calculated on the basis of the densities of the lumps on either end of the path, and includes the effects of any phase suction options. For homogeneous tubes, the factor UPF decides the weighted fraction for this density as it does for the frictional pressure drop correlations. In *all* connectors, the effective density is always the average of the lump densities—any other weighted average can cause numerical instabilities for two-phase flow. This means that UPF is ignored in homogeneous STUBE connector calculations for body forces, constituting the only deviation in meaning between tube and STUBE factors. If twin paths are used and if they are not currently homogeneous, then the density is reasonably constant for each phase. However, for stability, the uphill void fraction is used when calculating body force terms on each phase.

The use of average density for homogeneous STUBEs can have other modeling implications. To illustrate these implications, assume that an STUBE carrying liquid from an upstream lump flows into a plena containing gas or two-phase fluid. Even though the STUBE "carries" liquid, the effective density is somewhat less than pure liquid due to the averaging of the densities. Thus, the body force on that STUBE is less than might be expected. To overcome such cases where the user wishes to use specifically use only the inflowing state, an extra junction and path could be added at the outlet of the connector, with the junction having the same elevation as the outlet.

4.7.10.2 Referencing Body Force Options in Logic Blocks

CX, CY, and CZ may be used as variables within logic blocks in the same manner as TL and other lump descriptors (e.g., [fsmn.]CXn).

ACCELX, ACCELY, and ACCELZ may be used as variables within logic blocks in the same manner as ABSZRO and other global control constants (e.g., simply ACCELX).

The user should *not* reset these values in FLOGIC 1 or in VARIABLES blocks.

4.7.10.3 Gravity and General Acceleration Examples

This section contains simple examples of body forces due to both gravity and general accelerations. Refer to Section 3.11 for input formats. To summarize the rules previously described, either use a positive value of gravity and elevations for lump coordinates, or input ACCELX/Y/Z *as the direction in the (x,y,z) frame in which the whole system is accelerating.*

Gravity Example—In the following example, a vertical line is built whose positive flow direction is upwards. The line is 10 meters long, and is broken into 10 segments built using junctions and STUBE connectors. Note that lump 1 (not shown) has an elevation of 0.0:

```
HEADER CONSTANTS DATA, GLOBAL
     ACCELZ     = 9.81
     . . .
HEADER FLOW DATA ...
     . . .
M LINE,1,D,10,10,1
     DHS=0.01, TLENT=10.0,              NSEG=10
     LU=JUNC,  PA=STUBE,    CZ=1.0,     CZINC=1.0
```

General Acceleration Example—In the following example, a loop is built in the shape of a perfect square with sides equal to 10 feet (see Figure 4–18). The first leg extends 10 feet from the origin in the X direction from the origin, and is modeled by a single tube. The second leg bisects the positive Y and Z axes, and is modeled with a single STUBE connector. The third leg returns to the Y-Z plane, and is modeled with a center-differenced LINE using 10 segments. The last leg returns to the origin, and is modeled with a MFRSET pump (recall that there is no relationship between path length and the difference between lump coordinates).

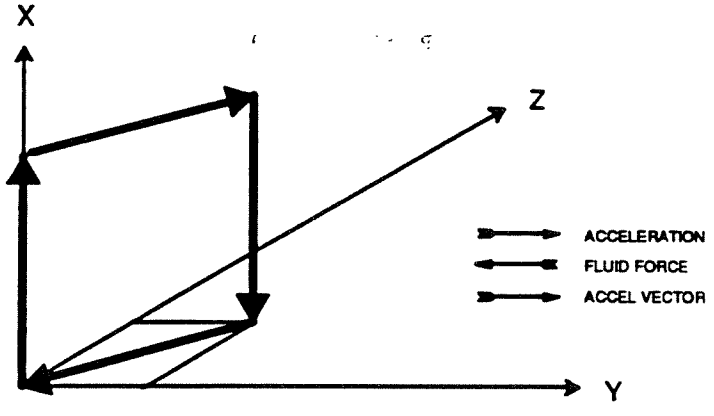


Figure 4–18 Orientation of Sample Loop

Initially, the spacecraft containing the loop is not accelerating. At time zero, the spacecraft fires a rocket and accelerates at 0.01g in the Y direction for 1 minute. Note that a vehicle acceleration in the Y direction results in a force on the fluid in the negative Y direction. ACCELY is the vector representing the vehicle acceleration in the lump coordinate system. A partial list of inputs is provided below:

```
HEADER CONSTANTS DATA, GLOBAL
     ACCELY     = 0.0
     . . .
HEADER OPERATIONS DATA
     . . .
     CALL FASTIC
     ACCELY     = 0.01*32.2*3600.0*3600.0      $ 0.01G
     TIMEN      = 1.0/60.0                     $ ONE MIN
     CALL FWDBCK
C
HEADER FLOW DATA ...
     . . .
LU JUNC,1                $ CORNER AT ORIGIN (DEFAULT)
PA TUBE,1,1,2
LU DEF,CX=10.0
LU JUNC,2                          $ CORNER (10,0,0)
PA CONN,2,2,3,DEV=STUBE,TLEN=10.0
LU DEF, CZ      = 10.0/1.414
     CY         = 10.0/1.414
LU JUNC,3                          $ CORNER (10,7.07,7.07)
M LINE,1,C,30,30,3,4 DHS=0.01, TLENT=10.0, NSEG=10
     LU=JUNC,   PA=STUBE,
     CX=9.5,    CXINC=-1.0      $ NOTE INIT. CX
LU JUNC,4,CX=0.0                   $ CORNER (0,7.07,7.07)
PA CONN,4,4,1,DEV=MFRSET,SMFR=100.0
     . . .
```

4.7.11 Variable Volume and Compliant Tanks

As the default condition, the walls of control volumes (tanks) are assumed to be fixed and infinitely rigid. Such an assumption is usually adequate for most analyses, but prohibits the modeling of many systems and devices. Fortunately, the user can override this assumption using either or both of two methods:

1) the control volume may expand or contract as a function of time, or

2) the control volume may stretch or shrink in response to pressure changes.

The ability to alter the size of the control volume allows the user to better simulate such diverse devices as diastolic pumps, pistons, and accumulators and such diverse phenomena as gas inclusions and separated phases.

4.7.11.1 Definitions

Tank walls may expand or contract over time if the *volume expansion rate* is nonzero. This growth rate, designated VDOT, is defined as:

$$\Psi = \frac{dV}{dt}$$

where:

Ψ ..... Volume expansion rate (VDOT)
V ..... Tank volume (VOL)
t ..... Time

The units of VDOT are volume/time; VDOT is positive for expansion and negative for contraction. The default is VDOT=0.0. Note that tank volumes may not be negative; a continuous negative VDOT will result in a diminishing time step as the volume approaches zero.

Tank walls may also be given a *compliance,* meaning they will stretch elastically if the pressure rises, and will shrink if the pressure drops. Tank wall compliance, designated COMP, is defined as:

$$C = \frac{1}{V}\frac{dV}{dP}$$

where:

C ..... Compliance (COMP)
P ..... Tank pressure (PL)

Compliance has units of inverse pressure (1/psi or 1/Pa, depending on the master model unit system). Compliance is the inverse of stiffness or volumetric spring rate; the higher the compliance, the softer the tank wall. A zero compliance (the default) corresponds to an infinitely stiff

tank wall, which is the default assumption. Negative compliances are legal and have even found some modeling uses, but they should be used with caution since they are can easily cause instabilities.

A tank with both nonzero COMP and nonzero VDOT will grow or shrink with both time and pressure changes:

$$\Delta V = \Psi \cdot \Delta t + C \cdot V \cdot \Delta P$$

A stiff-walled tank (small COMP) will be dominated by the VDOT term, growing or shrinking with time relatively independent of pressure changes. Conversely, a soft-walled tank (large COMP) may stay the same size while the pressure builds or drops according to VDOT (i.e., the tank wall may stretch to accommodate a negative VDOT). As an aid in thinking about the interactions between the two options and the modeling choices between them, VDOT terms may be thought of as a forcing function or as an action, while COMP terms may be thought of as a response function or as a reaction. For this reason, VDOT terms should be used carefully, while positive COMP terms are more innocuous and usually help smooth the system response.

Thermodynamically, a growing control volume performs work on its environment, whereas a shrinking volume absorbs work energy from its environment. For example, the temperature of a gas will rise as it is compressed, and will drop as it is expanded. This phenomena is correctly simulated in FLUINT whether the volume change is due to time or pressure or both.

4.7.11.2 Referencing COMP and VDOT in Logic Blocks

COMP and VDOT may be used as variables within logic blocks in the same manner as TL and other lump descriptors (e.g., [fsmn.]COMPn). The user should *not* reset these values in FLOGIC 1. Also, since tanks are treated like junctions in FASTIC, both of these factors are ignored in FASTIC with the exception that tank volumes are updated at the end of FASTIC for nonzero COMP to reflect overall pressure changes. Therefore, the user should temporarily set COMP to zero before calling FASTIC if he does not wish the volume to be affected by the call.

Several simulation routines assume control of the VDOT and COMP terms, including NONEQ1, BELACC, MIXGAS, and NCGBUB. When using those routines, the user should avoid altering these values.

4.7.11.3 Modeling with Volume Rates

Volume expansion rates can be used to simulate pistons, diastolic pumps or compressors (such as the mammalian heart), and accumulators that are squeezed or expanded. Because the system response can be very sensitive to changes in VDOT values, the following rules should be obeyed:

1) Avoid perpetually negative VDOT values—tank volumes must be positive.

2) Make sure the fluid has some where to come from or go to, such as a plenum, a compliant tank, or the compressibility of the fluid itself.

3) Do not change the value suddenly or extremely. Pressure spikes and notches will result when VDOT is changed on a liquid-filled tank; the adjacent flowrates must suddenly change to accommodate the new mass balance. Sinusoidal fluctuations in the volume can be absorbed, sawtooth fluctuations (discontinuous VDOT over time) will probably cause problems. In both cases, constraining the time step to a fraction of the period should be considered mandatory.

4.7.11.4 Modeling with Compliances

Compliances can be used to model flexible lines and nonrigid containers, liquid compressibility, noncondensible gases, and bladder accumulators. Adding even a small compliance to all tanks will result in smoother hydraulic responses and therefore less problems with user logic changes and real transient effects. *Small* compliances should be considered mandatory for hard-filled capillary tanks, hard-filled tanks next to control valves, and tanks with sudden VDOT or QDOT changes. (Refer also to the COMPLQ routine in Section 6.9.6.7).

Pipe Wall Flexibility—The spring rate inherent in the wall material of pipe or vessel can be modeled with a compliance. While flex hoses and rubber lines are obviously flexible, even metal pipes have some flexibility. For thin-walled pipes with small deflections that are within the elastic range, the compliance is constant:

$$COMP = \frac{D}{E\,t}$$

where:

D ..... Pipe diameter; the inner diameter (DH) is adequate
E ..... Young's Modulus or stiffness of wall material (units of pressure)
t ..... Pipe wall thickness (same units as D).

Gas Compressibility—Bubbles of noncondensible gases and gas in bladder accumulators can be simulated using compliances and assuming a perfect gas that is in thermodynamic equilibrium with the working fluid:

$$COMP = \frac{V_g}{P\,V_t}$$

where:

V_g ..... Gas volume
P ..... Tank and gas pressure, *in absolute units* (PL)
V_t ..... Tank volume (VOL)

Note that the fluid network does not include the gas itself, only its effect on the softness of the tank wall.

For small gas bubbles, the ratio V_g/V_t approaches the void fraction. For example, to add a small bubble of gas (say void fraction of 0.1%) in every tank to smooth the system response*, set COMP to 0.001 divided by a characteristic system pressure (say 14.7 psia) in an LU DEF subblock (see Section 3.11):

```
LU DEF, COMP    = 0.001/14.7            $ ENGLISH UNITS (1/psia)
```

To model a gas bladder accumulator with fixed total volume VOLACC, specify a tank of volume VOL (such that the volume of the gas in the accumulator is then VOLACC-VOL) and add the following logic to FLOGIC 0:

```
COMPn       = (VOLACC-VOLn)/(SNGL(PLn-PATMOS)*VOLn)
```

where n is the tank identifier. Additional logic may be necessary to ensure that VOL is less than VOLACC, or that a minimum compliance is maintained. The above formula assumes the bladder is very compliant compared to the gas; complex compliances are discussed below.

For very large gas bubbles, fast transients, or otherwise relatively slow heat transfer between the gas and the working fluid, the assumption of thermal equilibrium (constant temperature) may be replaced with one of constant entropy (i.e., a fast but reversible compression or expansion of the gas), resulting in a somewhat smaller compliance:

$$COMP = \frac{V_g}{\gamma P V_t}$$

where:

γ ..... Ratio of specific heats (C_p/C_v) for gas in bubble

Similarly, the compressibility of adjacent vapor (for hard-filled tanks next to a capillary device) or any other fluid may be simulated without assuming a perfect gas:

$$COMP = \frac{V_g \zeta}{\varrho_g V_t}$$

where:

ϱ_g ..... Density of the adjacent gas or fluid

ζ ..... The absolute value of the partial derivative of density with respect to pressure at constant temperature (or constant entropy for quick changes) for the adjacent gas or fluid (may be found using property routines for standard library fluids)

To model an adjacent vapor volume where heat and mass transfer between the liquid and vapor phases must be included, refer to the NONEQ0/1 routines in Section 6. See also the simulation routine NCGBUB for a more complete treatment of a noncondensible gas phase in thermal equilibrium with (and in contact with) a subcooled liquid phase, and the simulation routine BELACC for simulation of vapor adjacent to but separated from liquid.

* In many systems, tiny gas bubbles are purposely introduced to avoid pressure surges.

Liquid Compressibility—FLUINT normally assumes an incompressible liquid state. If desired, the compressibility of the liquid can be modeled using compliances as follows:

$$COMP = \frac{1}{K_B} = \frac{1}{F\rho a^2}$$

where:

K_B ..... Adiabatic bulk modulus of liquid (units of pressure)
F ..... unit conversion factor: 1.0 for standard SI units, 1.665E-11 for standard ENG units (ft, hr, lb_m, psi)
ρ ..... liquid density
a ..... Speed of sound in liquid (use VSOSF routine unless using 9000 series fluids)

The COMPLQ routine in Section 6.9.6.7 automates the addition of this term.

NOTE: FLUINT does not track shock waves, and is not strictly intended to be used to simulate waterhammer or column separation events. The solution method used specifically avoids resolution of such "fast transients" in order to allow larger time steps. When the code is used on systems exhibiting such behavior, the pressure spikes and notches will be smoothed making the code nonconservative for pipe-burst and similar studies. However, used properly with a sufficiently small time step (DTMAXF), FLUINT has compared favorably with codes that are specifically intended to simulate waterhammer. Refer to Section 6.9.6.7 for guidelines and support routines.

Adding Compliances—If there is more than one reason for a tank to be compliant, a total effective compliance can be easily calculated. Thus, all of the above phenomena (tank wall flexibility, liquid or gas compressibility) can be summed together and simulated simultaneously.

The formulas differ according to whether the compliances are in series or parallel. A gas bubble *inside* a rubber-walled tank is an example of parallel compliances, since there is a volume change associated with *each* cause. Any number of parallel compliances can be added algebraically to yield a single effective compliance:

$$COMP_{eff} = COMP_1 + COMP_2 + COMP_3 + \cdots \quad \text{(PARALLEL)}$$

A gas bubble *behind* the wall of a rubber-walled tank (e.g., a gas-bladder accumulator) is an example of series compliances, since there is *one total* volume change associated with *both* causes. The inverse of any number of series compliances can be added algebraically to yield the inverse of a single effective compliance:

$$COMP_{eff}^{-1} = COMP_1^{-1} + COMP_2^{-1} + COMP_3^{-1} + \cdots \quad \text{(SERIES)}$$

Combined series and parallel compliances can be calculated using a spring rate or electrical resistance analogy (such as is used in SINDA!). For example, assume there is gas both behind and inside a bladder. The compliance of the bladder is added to the compliance of the gas behind the bladder using the series formula, and then this effective compliance is added to the compliance of the gas inside the tank using the parallel formula.

When modeling spring-like devices such as metal bellows, it is important to note that compliances are analogous to the "kx" term in the spring equation $F = k(x-x_o) = kx - kx_o$. The term driving the spring to the return position or *rest state*, kx_o, is *not* included in the compliance term.

This driving term must be added separately via VDOT values, or perhaps even the HC of adjacent tubes or STUBE connectors. Alone, compliances can only be used to model such spring-like devices if the rest state corresponds to the initial condition. See also the simulation routine BELACC in Section 6 for a more complete treatment of a bellows accumulator.

4.7.11.5 Compliances as Smoothness and Safety Measures

Strictly, compliances should be used to represent real phenomena, such as gas inclusions and elastic walls, and *all* pipes and containers exhibit some compliance even if this is limited to liquid compressibility and the flexibility of a metal pipe. Unfortunately, severe transient events (such as rapid changes in pumps, valves, VDOT, and QDOT values) coupled with the assumed incompressibility of liquid can result in unrealistically large pressure spikes and notches in hard-filled portions of the network. Therefore, adding extra "artificial" compliance to tanks will make the program much more rugged. Even errors in program use (such as situations where the fluid has nowhere to flow) are easier to detect, and coarse changes made in user logic blocks are more tolerable. For these reasons, *small* compliances should be considered mandatory for hard-filled capillary tanks, hard-filled tanks next to control valves, two-phase tanks that can become hard-filled, and tanks with sudden VDOT or QDOT changes. Refer also to the COMPLQ routine in Section 6.9.6.7.

However, note that an unrealistically large compliance can give misleading results by allowing the tank to stretch like a balloon (the simulation would be correct, but the model would be inappropriate). ***Compliances are inherently small numbers***; typical values might be 1.0E–4 in English units, and 1.0E–8 in SI units.

4.7.12 Spatial Accelerations and the AC Factor

Each path in FLUINT has one characteristic flowrate. Therefore, each tube or STUBE has one characteristic velocity. In many cases, however, the "real" velocity on one end of a path is really never equal to that on the other end, even at steady-state. This can be caused by flow area changes (e.g., reducers and diffusers) or density changes (due to heating or cooling), and may be caused by other modeling assumptions, as will be shown below. In any case, there is an associated pressure gain or drop with such a velocity gradient. This pressure gradient is proportional to the *spatial acceleration* of the fluid: a driving force is required to accelerate or decelerate the fluid within the path. This is a reversible effect, meaning that *if the flowrate reverses, a drop becomes a gain and vice versa.* The AC (Area Change or ACceleration) factor models such reversible losses.

All AC factors are based on *defined* inlet and outlet conditions, and do not change if the flowrate reverses. A positive value therefore implies a pressure gain in the positive flowrate direction, and a loss in the reverse direction. (By comparison, FC factors are always negative and always cause a loss in any direction—they are *ir*reversible.)

Area Changes—For area changes in incompressible flow, AC is only a function of fluid density and the *defined* inlet and outlet flow areas AF_i and AF_j, respectively:

$$AC = \frac{1}{2\varrho}\left(\frac{1}{AF_i^2} - \frac{1}{AF_j^2}\right) \qquad \text{(AREA CHANGE, INCOMPRESSIBLE)}$$

The reader might notice that this is actually a form of Bernoulli's equation, which is an energy relationship rather than a momentum relationship. *Only in incompressible flows can the AC for area changes be known exactly* without a detailed profile of flow area vs. axial distance and a few other simplifying assumptions. Even incompressible AC factors could not be calculated exactly without the use of energy relationships such as Bernoulli's.

Even though the user might calculate AC given two flow areas, only one AF and DH are used to describe the tube or STUBE for the purposes of friction calculations (i.e., FC and FPOW) and for heat transfer calculations (if the path is used in an HTN, HTNC, or HTNS tie). The user must decide what values of AF and DH are applicable for such calculations. For heat transfer purposes, the AF and DH could be based on an average cross section. For friction purposes, however the user either should err on the small side to reflect the added irrecoverable losses associated with an area change, or should add a LOSS2 connector in series. To err on the small side, an inverse geometric average of the inlet and outlet flow areas may be desirable:

$$AF_{eff} = \left(\frac{1}{2}\left(\frac{1}{AF_i^2} + \frac{1}{AF_j^2}\right)\right)^{-0.5} \qquad \text{(EFFECTIVE AREA)}$$

$$DH_{eff} = \sqrt{\frac{4}{\pi} AF_{eff}} \qquad \text{(EFFECTIVE DIAMETER)}$$

Any desired differences between the heat transfer geometry and the friction geometry can be accommodated via the area fractions or tie duplication factors. Also, the tube or STUBE connector can be frictionless by specifying IPDC=0 and FC=0.0 as long as AC and FK are never both

zero. This assumption implies an ideal reducer or diffuser. In that case, the AF and DH are only relevant for HTN, HTNC, and HTNS ties, although the AF of a tube is still used in calculating its inertia.

The internal throat area, AFTH, is independent of these calculations. (In fact, the only current use for the AFTH factor is in the choking detection routines CHKCHL and CHKCHP.) In other words, no assumption is made with regard to where within the path the throat exists.

Density Changes—AC may also be used to represent spatial accelerations in homogeneous paths due to density changes between the inlet and the outlet. Such spatial accelerations are significant in evaporating or condensing two-phase ducts. Mathematically, spatial acceleration is due to a difference in momentum flux. Conceptually, it can be thought of as the pressure loss needed to accelerate a slow moving liquid into a fast moving vapor, or the recovery of pressure from condensing high speed vapor:

$$AC = \frac{1}{AF^2}\left(\frac{1}{\varrho_i} - \frac{1}{\varrho_j}\right) \qquad \text{(DENSITY CHANGE ONLY, HOMOGENEOUS FLOW)}$$

where ϱ_i and ϱ_j are the densities at the *defined* inlet and outlet, respectively.

Combined Area and Density Changes—When both area changes and density changes occur (e.g., a shear flow condenser), the AC factor cannot be calculated exactly without much more detailed information or empiricisms. In practice the area change contribution is usually negligible compared to the density change contribution. If this is not the case, then the two effects may be considered simultaneously in a first-order approximation by algebraically summing the two AC contributions, using an effective area in the density change equation and an effective density in the area change equation:

$$AC = \frac{1}{2\,\varrho_{eff}}\left(\frac{1}{AF_i^2} - \frac{1}{AF_j^2}\right) + \frac{1}{AF_{eff}^2}\left(\frac{1}{\varrho_i} - \frac{1}{\varrho_j}\right) \qquad \text{(DENSITY AND AREA CHANGE, APPROXIMATION)}$$

where the effective flow area might be calculated given the previous formula, and one possible equation for the effective density is:

$$\varrho_{eff} = \left(\frac{1}{2}\left(\frac{1}{\varrho_i} + \frac{1}{\varrho_j}\right)\right)^{-1} \qquad \text{(EFFECTIVE DENSITY)}$$

If the area change is indeed gradual, then a better approximation for the combined AC factor is:

$$AC = \left(\frac{1}{\varrho_i AF_i} - \frac{1}{\varrho_j AF_j}\right)\left(\frac{2}{AF_i + AF_j}\right) \qquad \text{(DENSITY AND GRADUAL AREA CHANGES)}$$

The above equation has the benefit of reducing to the correct momentum flux equation in the limit case where the flow areas are equal, but unlike the summation method it does not reduce to Bernoulli's equation in the limit case where densities are equal. The latter limit is more likely to occur during a simulation.

Automatic AC Calculations in LINE and HX Macros—The AC factor is automatically calculated in macros generating duct models (LINE and HX, discussed below). Otherwise, it is left as zero. The user forfeits the ability to independently specify the AC factor when using a LINE or HX macro, although the AC factor for tubes may be modified in FLOGIC 1 in all solution routines except FASTIC. For twinned tubes and STUBE connectors, the FG coefficient follows the above rules.

In homogeneous LINE and HX macros, a formula similar to the above density change formula in invoked, although other factors such as mass flowrate gradients due to compressibility or side flow passages are also included. If the paths are twinned, then the basis for the internal algorithm is derived from the above area change formula, since each twin has nearly constant density but occupies a varying portion of the flow passage. Furthermore, if paths are twinned the related FG coefficient is calculated, accounting for the small momentum transfer associated with phase change.

An AC factor with the same sign as the flowrate FR means that the fluid is decelerating, which can actually result in a pressure gain instead of a drop. While this is physically realistic, the assumption of instantaneous reaction in STUBEs combined with decelerations can cause the flowrate to fluctuate wildly and reverse suddenly. Such behavior is often seen in rapidly condensing LINE and HX macros built with STUBEs. To avoid such instabilities, the user should add more reality to the model in the form of head losses on the high-velocity vapor side. For example, FK=0.5 might be placed on the inlet path of a condensing duct to represent entrance losses in the manifold or header. If tubes are used, then the time step may be limited by stability for analogous reasons. If so, then the time step status line will indicate a "STABILITY LIMIT."

In LINE or HX macros where the evaporation or condensation rate is severe, the fluid might boil or condense within a distance shorter than the discretized segment length. In such cases, the program will have difficulty converging or will require small time steps because of the sharp gradients. These gradients are particularly troublesome in condensers due to the spatial deceleration, but can be troublesome in any case because they affect heat transfer calculations and frictional calculations as well. Either more spatial resolution is required, or the user might approximate the heat transfer process using a heater junction. Refer to Sample Problem E (specifically, the steady-state initial conditions for the abbreviated model) and Sample Problem F (specifically, model of the internal heat exchanger) for examples.

If the user elects to perform spatial acceleration calculations when the cause is density changes, extreme caution should be used because of the potential for serious numerical problems. The above formulae do not take into account transient lags in velocity in ducts modeled with tanks, nor does it include the effects of side passages. Both effects are automatically included in LINE and HX macros.

Spatial acceleration may also result from radial blowing or suction, causing an axial velocity gradient. In this case, the difference in momentum flux is due to differences in the direction and magnitudes of the inflowing and outflowing momentum vectors. One example occurs in the modeling of heat pipe type channels. In such channels, evaporation at the wall injects mass radially into the flow, forcing an acceleration, while condensation extracts mass radially, decelerating the flow. In either case, the flowrate at one end of the channel is zero. If the injection or extraction is uniform, the flowrate along the length of the channel is linear, as shown in Figure 4–19.

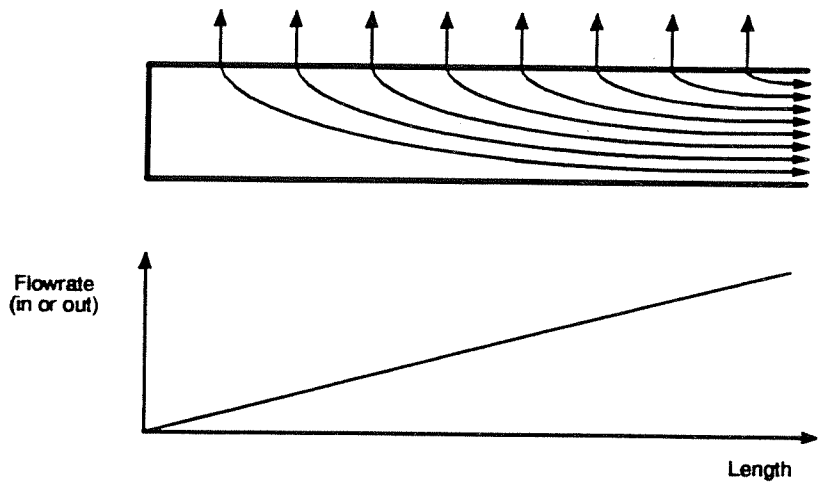


Figure 4–19 A Passage with Uniform Radial Injection/Extraction

To model the frictional characteristics of such as passage, a common heat pipe "half-length" assumption is used: the full flowrate is assumed to act over half the length. To include the acceleration effects (the fact that the fluid being injected or extracted radially does not contribute the axial momentum flux):

$$AC = \frac{1}{\varrho\, AF^2} \qquad \text{(Suction, extraction)}$$

$$AC = \frac{-1}{\varrho\, AF^2} \qquad \text{(Blowing, injection)}$$

Notice that the sign is determined by whether the flowrates are positive for suction or for blowing, but do not change if the flow reverses (i.e., blowing becomes suction). In other words, if the flowrate is zero at the defined upstream end, the sign of the AC factor is negative since only the downstream flux term remains, and vice versa. Sample Problem G contains examples of both side flow passages and the equivalent AC factors needed to account for them when the model is simplified.

The above modeling method assumes one lump and one path are used to represent the whole segment. If it is desired to break the segment up into smaller pieces, the equations become more complicated. For such cases, the use of a LINE or HX macro is recommended.

4.7.13 Flow Regime Mapping and Related Pressure Drop (IPDC = 6)

If IPDC, the frictional pressure drop correlation identifier for tubes and STUBE connectors, is set to zero, then the user assumes responsibility for calculating and updating FC and FPOW, which are described in Section 4.7.3.2. Furthermore, if the paths are twinned, the user assumes responsibility for calculating and updating FD as well.

However, such usage is the exception rather than the rule. Instead, IPDC is nonzero by default. This invokes automatic calculation of FC and FPOW for single phase flow based on Moody chart friction factors. In the two–phase regime, different correlations will be applied depending on the value of IPDC. For options 1 through 5, these correlations are described in Appendix A.

If IPDC = 6, a best–guess two–phase flow regime will be predicted and the pressure gradient will be estimated on the basis of that regime. If UPF is not 0.0 or 1.0, each tube or STUBE connector will be described by distinct flow regimes at both ends. The PTHTAB output routine prints the current flow regimes at each end of the tube or STUBE connector if IPDC=6 and the flow is two–phase. If the flow is single–phase, the Reynolds number is printed instead.

The special case of IPDC = 6 is the subject of this section, which describes the simplified flow regimes used internally, the rationale for choosing between them, and the implications of that choice on the frictional pressure drop. While this option may be used for single homogeneous paths, it is the default option for slip flow (twinned) paths. Hence, in many ways this discussion is a prelude to the next section (Section 4.7.14) detailing slip flow modeling.

The principal reference for this section is the Design Manual for Microgravity Two–Phase Flow and Heat Transfer (AL–TR–89–027), October 1989, by C.J. Crowley and M.G. Izenson. Note that the methods used within FLUINT deviate somewhat from those described in that reference, mostly in (1) the addition of regime hysteresis, (2) the extension of these methods to unsteady and diabatic flows, (3) the preclusion of stratified flow if the liquid can reach the top of the pipe by surface tension alone, and (4) different methods for predicting pressure drop in all regimes except stratified. With respect to slip flow modeling, the differences are even greater.

4.7.13.1 Flow Regime Descriptions and Distinctions

Four generalized (simplified) regimes are recognized, as illustrated in Figure 4–20: bubbly, slug, annular, and stratified. The first two are considered "dispersed," and the latter two "separated." The distinction between regimes is based (1) on the liquid and vapor mass fluxes, (2) on the void fraction, (3) on the hydraulic diameter of the line—assumed nearly circular, (4) on the magnitude of a body force (or acceleration) vector and its orientation with respect to the duct, (5) on fluid properties such as densities, viscosities, and surface tension, and (6) in the event no clear determination can be made, on previous flow regimes (i.e., regime boundaries exhibit hysteresis).

Bubbly flow occurs at the extremes of low gravities, high liquid mass fluxes compared to the vapor flux, and low void fractions (less than about 0.46), and is characterized by small vapor bubbles entrained in liquid. If the bubbles coalesce due to increased accelerations, decreased liquid mass flux, or increased void fraction, then the slug flow regime will appear. The slug flow

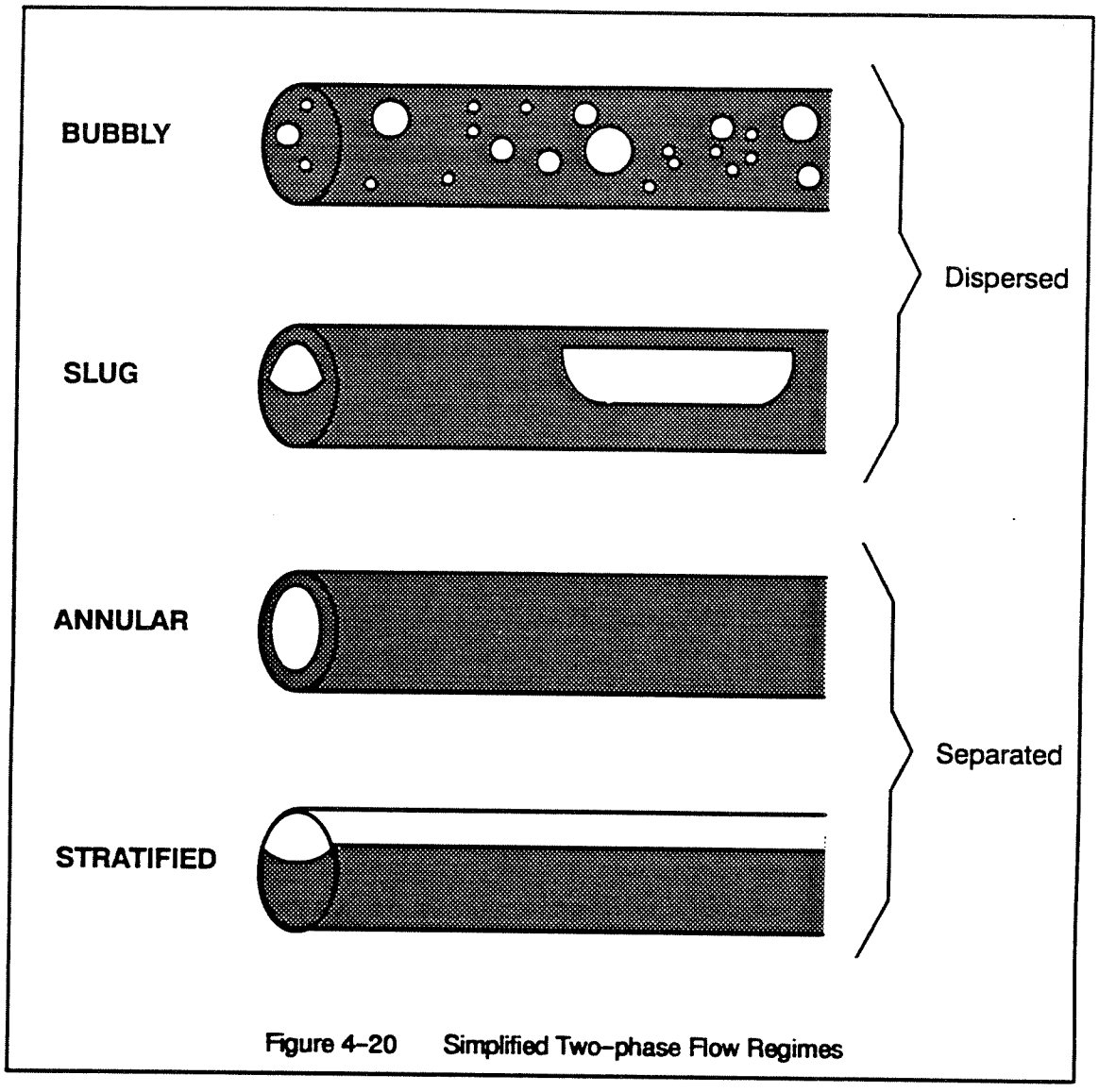


Figure 4-20 Simplified Two-phase Flow Regimes

regime is typified by large bubbles that nearly span the diameter of the tube, but which are axially separated from each other by liquid. Both the slug and bubbly flow regimes are characterized by relatively little slip flow, approaching true homogeneous flow. In both cases, predicted pressure drops are based on the McAdam's formulation for homogeneous flow (IPDC=1). These two regimes are therefore identical for single homogeneous paths, but they behave differently if slip flow is modeled (see Section 4.7.14).

The annular regime may result if the void fraction continues to grow (above about 0.76), or if the liquid flows downhill, or if there is high enough vapor flux to sustain the uphill flow of liquid. This regime is characterized by a continuous vapor core surrounded and 'lubricated' by a continuous liquid annulus. In most two-phase systems, annular is by far the most common regime. When the regime is determined to be annular, the Lockhart-Martinelli (IPDC=2) correlation is used.

The stratified regime, characterized by liquid pooling in the bottom of the tube, results if either the vapor mass flux or the liquid fraction is low enough, or the gravity high enough (and the flow is not vertically upward). The stratified regime cannot exist in microgravity. The methods used to predict pressure gradient involve predicting the height of liquid and the fractions of each phase in contact with the wall, assuming a circular cross section (per the method of Taitel and Dukler, 1976). Unfortunately, this model is highly sensitive to void fraction, and because the stratified regime typically exhibits the greatest degree of slip, the error in a homogeneous approximation to void fraction can be significant. In other words, *pressure drops in the stratified regime are suspect if the default homogeneous options are used.* Typically, a homogeneous assumption results in overestimation of pressure drop for stratified flow, whereas if slip flow is modeled, the predicted pressure drop is usually lower than that of all other regimes for the same flow quality.

4.7.13.2 Regime Uncertainties: Simplification and Hysteresis

The four possible flow regimes are considered simplified because they represent the top level classifications for families of more detailed regimes. For example, the annular regime is usually subdivided into regimes such as "wispy annular," "misty annular," and "wavy annular." However, the exact distinction between these subdivisions quickly becomes subjective, and the differences between mechanistic descriptions become negligible.*

Even allowing for such simplifications, the term 'best–guess' was not used lightly in the opening discussion. The user should not have the impression that flow regime prediction is an exact science. Effects of heating, cooling, upstream conditions, and past history can all affect the local flow regime, and as noted above, the literature (which, as usual, is largely devoted to water) recognizes many more than the four regimes used in FLUINT. This regime predicting capability simply attempts to discern the major breakpoints in behavior on the basis of mechanistic principles.

For example, bubbly flow does not usually exist when the void fraction exceeds some critical limit. Beyond this critical void fraction, the bubbles would exceed the maximum packing and coalesce, and the flow transitions to the slug regime. Values for this critical limit can range from 0.4 to 0.52, and even this range should not be treated as 'gospel.'

Because of these uncertainties and because of the substantial differences in behavior between regimes, FLUINT employs substantial hysteresis in regime prediction for stability: *when in doubt, the last regime is favored over an alternative regime.* In effect, *all* documented uncertainties are treated hysteresis. While there is some physical basis for flow regime hysteresis, in reality not all uncertainty in regime determination can be attributed to hysteresis.

---

* Perhaps the greatest error in these simplified regimes is the lack of an entrained liquid phase within the vapor spaces of separated regimes. Inclusion of such detail would be nonsensical for a homogeneous model, and cannot be treated with much confidence in a more detailed slip flow model, largely because of the extreme sensitivity to upstream conditions (developing flow) and history. Furthermore, the low pressure gradients and low vapor fluxes characteristic of spacecraft thermal management systems do not justify the extra effort: entrainment is minimal. Another source of error in the simplification of the flow regimes is the lack of bubbles within the liquid spaces of slug, annular, and stratified regimes. Such effects occur at high fluxes and/or in nucleate boiling, neither of which are of great importance in the spacecraft industry.

When uncertainties were not quantified, hysteresis was created somewhat artificially. For example, the critical transition criterion for slug to annular is given in the aforementioned Design Manual as a void fraction of 0.76. (Actually, this is a bit high compared to other references, again underscoring the inherent uncertainty in the flow regime determination game.) In FLUINT, annular cannot exist when the void fraction is below 0.9*0.76, and slug cannot exist above 1.1*0.76.

From a user's perspective, this hysteresis means that different results may be achieved when starting from different initial conditions. While such behavior is certainly not new, it is perhaps more exaggerated with IPDC=6 than with other current causes of hysteresis. Different (but hopefully equally correct) answers are especially likely where flow distributions between parallel legs are calculated, and where the flowrates are based on pump curve matching, etc.

4.7.13.3 Interactions with the ACCEL vector

The body force or vehicle acceleration vector ACCEL now affects not only axial pressure forces but also flow regime determination as well when IPDC=6. Since ACCEL vector components (ACCELX, ACCELY, and ACCELZ) all default to zero, *flow regime predictions are based on microgravity conditions by default.* Any nonzero ACCEL components might have an effect on the predicted flow regime.

Such interactions are based both on the magnitude of the ACCEL vector and on its orientation with respect to the path vector. The latter is defined as the vector pointing from the coordinates of the upstream lump to the downstream lump. Recall that the path vector length does not have to match the input length, TLEN, of the tube or STUBE connector.

NOTE: When the path vector has zero length, the ACCEL vector is assumed to be perpendicular to it. This is important since all lump coordinates are zero (coincident at the origin) by default, and therefore all path vectors are zero by default. Thus, *the user can quickly assess the impacts of gravity on a horizontal system without having to input any lump coordinates at all* since any input ACCEL vector will be treated as perpendicular to *all* tubes and STUBE connectors if no lump (CX, CY, CZ) coordinates have been defined.

Irrespective of orientation, the magnitude of the ACCEL vector can affect the bubbly/slug transition as well as the stratified/other transition. If the magnitude of the body force is high enough, the bubbly regime will disappear in favor of the slug regime since bubbles will coalesce into slugs due to buoyancy forces. Stratified flow cannot exist at all unless the magnitude of the ACCEL vector is great enough.

Orientation with respect to the body force vector can be important, too. Separated flows (stratified and annular) usually cannot exist if the flow is rising relative to the body force (e.g., vertical upflow) since the liquid falls back on itself and the slug regime results. This *flooding* is less likely to occur at high qualities where the vapor shear prevents it.

4.7.14 Slip Flow Modeling Using Twinned Paths

By using a single path to model a passage in which two–phase fluid is flowing, the user is implicitly assuming that the flow is homogeneous, meaning that the velocities of each phase are equal. In fact, for most purposes homogeneous fluid flow is modeled as a single effective 'phase' having properties intermediate between pure liquid and pure vapor phases.

The assumptions of "one lump, one thermodynamic state" and "one path, one characteristic flowrate" are fundamental to FLUINT's lumped parameter network style. Actually, the latter statement should be revised to read "one path, one characteristic *velocity*." Other than that lexical change, slip flow modeling does not violate those assumptions. Just as the modeling of non-equilibrium (imperfectly mixed) two–phase control volumes with the NONEQ0/1 routines requires two control volumes, one for liquid and one for vapor, the modeling of unequal phase velocities requires two paths.

Before describing how slip flow can be modeled using FLUINT, some background information will be provided for those users unfamiliar with two–phase modeling. Flow regime mapping methods described in Section 4.7.13 are also a prerequisite for this section.

4.7.14.1 An Introduction to Slip Flow

In all *single* FLUINT paths, a homogeneous assumption is used, meaning that the vapor velocities and the liquid velocities are assumed equal: there is zero relative velocity or slip. With the homogeneous approximation, two–phase flow is modeled as the flow of a mixture of both phases—one momentum equation describes the entire path. This assumption is usually adequate and is both simple to implement and fast to execute. Because of this assumption, there is no difference between flow quality and thermodynamic quality. Thermodynamic quality is the fraction of vapor *mass* within a segment divided by the total *mass* in that segment: $M_v/(M_v+M_l)$. Flow quality is the ratio of vapor *mass flowrate* through a segment divided by the total *mass flowrate* through that segment: $FR_v/(FR_v+FR_l)$.

In reality, vapor usually moves faster than liquid, and sometimes even in opposite directions. A slip flow formulation takes this into account, using one momentum equation per phase (see subsection 4.7.3.2). Unlike homogeneous flow, with slip flow the thermodynamic quality is no longer the same as the flow quality. Conservation of mass dictates that flow quality must be the same (eventually) whether a homogeneous or slip flow formulation is used. However, the thermodynamic quality is no longer constrained by the homogeneous assumption: it becomes the new degree of freedom necessary to accommodate a new momentum equation. In other words, the thermodynamic quality and its manifestations, such as the density and void fraction, will vary as needed to balance the flow forces. Because vapor generally travels faster than liquid, *the predicted void fraction will be smaller with slip flow than with homogeneous flow at the same flow quality.* In other words, more liquid will reside in the line, and the thermodynamic quality will be smaller than the flow quality. In FLUINT parlance, if twinned paths are employed then the lump XL will no longer be an indication of the flow quality even at steady state. Figure 4–21 graphically illustrates this point for the stratified regime.

Because most pressure drop and heat transfer correlations are based on flow quality, slip flow and homogeneous formulations predict almost the same steady state as long as flow is co-current (see Sample Problem G); the local homogeneous assumption does not affect the overall

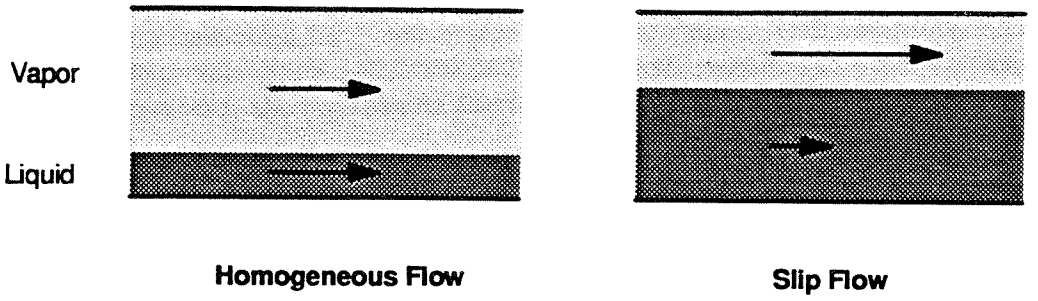


Figure 4–21 Homogeneous versus Slip Flow in Stratified Regime at Same Flow Quality

pressure drop and heat transfer rates. The major difference is the proportion of liquid and vapor in lines. For example, in annular flow a slip formulation predicts typically three to four times as much liquid will reside in a pipe compared to a homogeneous prediction. Of course, this amount is small to begin with, and so quoting a factor of three to four might be misleading.

In transients, the differences can be more dramatic, especially for separated flow regimes where vapor can shift quickly and liquid lags behind. As a specific example, a SINDA/FLUINT model was developed to predict the time it takes to clear a small tube of liquid by heating it, noting that much more liquid is displaced by generated vapor than is actually evaporated. The default homogeneous assumption resulted in a prediction of 8 seconds to clear the line, whereas allowing slip flow in the same model nearly doubled the duration of the liquid purge event. Since annular flow was quickly established, slip flow allowed the vapor to escape the tube without displacing as much liquid in the process.

This extra degree of modeling power does not come without its price. In addition to greater solution expense, a new layer of uncertainties is revealed. New parameters must be estimated, including (1) the frictional drag between phases, (2) the degree of sharing of inertia, also called added mass and virtual mass, (3) the apportionment of wall friction to each phase, and (4) the momentum transfer associated with phase change. In FLUINT terminology, these factors are called FD, AM, FC/FPOW, and FG, respectively. By default, FLUINT will estimate these factors automatically, which requires knowing the flow regime. Hence, flow regime mapping options (IPDC=6) are defaulted when specifying slip flow. Alternately, like almost all other SINDA/FLUINT options, knowledgeable users can calculate their own coefficients.

4.7.14.2 Twinned Paths: Concept and Usage

To model slip flow in a pipe, the user creates a *pair* of twinned tubes or twinned STUBE connectors that will together model *one* flow passage. Nominally, one will carry liquid and the other vapor. They start and end at the same two lumps, sharing one definition of positive flow-rate direction, diameter, flow area, wall roughness, duplication factor, etc. Although they are intimately interrelated, they are distinct paths that require distinct user identifiers.

To maintain commonality with single homogeneous paths, one of the twin paths is considered *primary* and the other *secondary*. When carrying two–phase flow, the primary path carries the liquid phase and the secondary carries the vapor phase. (This distinction is arbitrary, implying no favoritism to one phase or the other.) In FLOW DATA, the secondary path is named after the input keyword "TWIN = ," the presence of which causes two paths to be generated simultaneously.

When only one phase or the other is present in the line, only the primary path will be used even if the remaining phase is vapor. The secondary path is largely ignored until the absent phase reappears. The former mode, where both twins are active, is called the *slip flow mode*, and the latter mode, where only the primary path remains, is called the *homogeneous mode*. Refer to subsection 4.7.14.3 for a more detailed discussion of the mode transitions. Refer also to the GOHOMO and GOSLIP routines in Section 6, which give the user explicit control over the above modes.

Input—All paths may be twinned. However, slip flow modeling logic is only invoked for twinned tubes and STUBE connectors. Other device types (MFRSET, etc.) may be twinned, but the two nearly identical paths then operate independently. For example, if two MFRSET connectors are twinned, they will be identical with the exception that the first (primary) device will be initialized with STAT=DLS, and the second with STAT=DVS. No other logic or simulations are applied thereafter.

Subsection 4.7.3.2 describes the tube and STUBE governing equations, along with the definitions and descriptions of the relevant parameters. The rules governing the calculation of FD, FG, and AM are similar to those governing FC and AC. FD is automatically calculated if IPDC=6, otherwise it defaults to zero. FG, like AC, is only calculated automatically if the twinned paths were generated as part of a duct (LINE or HX) macro. Otherwise, FG defaults to zero and may be user specified. AM is always calculated for tubes. Figure 4–6 details the sequence of updating these parameters and the appropriate logic block to use for modifying them.

The default automatic calculations of FD, FG, and AM meet a special requirement: with the exception of the dissipative FG term, when the two equations for both phases are summed, both the FD and AM terms sum to zero. The FG terms would also sum to zero if an average interface velocity were used instead of the dissipative formulation used in FLUINT. Also, the total pressure forces sum to PL_i–PL_j since the fraction of that force on each phase is simply the volumetric fraction.

Use IPDC=6 for twinned paths, otherwise FD will not be calculated automatically and defaults to zero. Zero FD may cause instabilities because it is nonphysical. If IPDC is never defaulted in a PA DEF subblock nor otherwise stated explicitly, then the IPDC for all twinned tubes and STUBE connectors will default to 6, whereas the normal default is 1. ***Explicit input of IPDC=6 is recommended for twinned paths.***

Note that *twinned paths cannot use the phase suction options*. Any such inputs (e.g., "STAT=...") will be accepted but ignored.

All HTN, HTNS, HTUS, and HTNC ties (including those generated by macros) must refer only to the ID of the primary tube or STUBE of a twinned pair. Involvement of the secondary path is implied, but explicit reference to it in a tie is not allowed. This treatment is compatible with the homogeneous mode, in which the secondary twin is largely ignored.

Output—The PTHTAB output routine will print flow regimes or Reynolds numbers for tubes and STUBE connectors. When a pair of tubes or STUBE connectors is in the slip flow mode, each will be listed separately as a single phase duct, where the Reynolds number is given as if the flow existed alone in the duct (i.e., based on superficial velocity). In this mode, the primary path ID is printed with an "L" suffix, and the secondary path ID is printed with a "V" suffix. When they are in the homogeneous mode, the secondary path is omitted, and the primary path ID is printed with an "H" suffix.

Despite input values of UPF, for stability all twinned paths internally use UPF=1.0 while in the slip flow mode. This means that of the two columns available in PTHTAB for up– and downstream flow regime or Reynolds number, only the one corresponding to the current upstream lump will be used. An exception is countercurrent flow, where a weighted combination of each state is used. When the pair enters the homogeneous mode, normal usage of UPF resumes.

An output routine similar to PTHTAB is also available that treats twinned pairs as single effective paths, printing out one pair per line. This routine, TWNTAB, ignores all single paths and unlike PTHTAB, prints flow regime and effective flow quality for all twins. In TWNTAB, when a twinned pair is in the slip flow mode, the primary path ID is printed with an "L" suffix, and the secondary path ID is printed with a "V" suffix. When they are in the homogeneous mode, and the primary path ID is printed with an "H" suffix, and the secondary path ID is printed with an "X" suffix signalling that it is currently being ignored.

TWNTAB has exactly the same argument as PTHTAB: the fluid submodel name or 'ALL' in single quotes. The following example demonstrates recommended usage. Consider calling both PTHTAB and TWNTAB when twinned paths are used:

```
HEADER OUTPUT CALLS, BOBSEY
         CALL LMPTAB('ALL')
         CALL PTHTAB('ALL')
         CALL TWNTAB('BOBSEY')
```

References in Logic Blocks—Each member of a twinned pair may be referenced in any logic block at any time, whether they are currently in the slip flow mode or the homogeneous mode. However, certain parameters such as DH and TLEN are shared by each pair, meaning that the same value is used for both paths. These parameters are: DH, TLEN, AF, AFTH, WRF, UPF, IPDC, DUPI, and DUPJ. While any of these values may be referenced or altered for either path, *changes to these values will be ignored for the secondary path and will be overwritten by the corresponding value for the primary path during each solution step.* In other words, simply use the primary path ID in referencing any of these parameters, and then they can be accessed and altered in any manner legal for single homogeneous paths. Rephrasing, the secondary paths will contain the same values as the primary paths, and hence can be accessed equivalently, but any changes to these values should reference the primary path.

For example, consider that tubes #34 and #1034 are paired, with #34 as the primary twin:

```
HEADER FLOW DATA, DUBLMINT ...
          ...
PA TUBE,34,3,4,    TWIN = 1034 ...
          ...
HEADER FLOGIC 0, DUBLMINT
          IF(TLEN1034 .GE. 1.0)THEN  $ REF TO SECONDARY TUBE OK
               TLEN34    = 0.5       $ CHANGE TO PRIMARY TUBE OK
               DUPJ1034 = 0.0        $ DOES NOTHING: OVERWRITTEN LATER
          ENDIF
```

See also the Section 6 describing GOHOMO and GOSLIP, auxiliary routines that allow the user to explicitly determined whether the tubes and STUBE connectors may slip, or whether they should be confined to the homogeneous mode.

Like CAPIL and CAPPMP, phase suction options are used internally to model slip flow for active twins. Therefore, any calls to CHGSUC will be later overwritten and therefore effectively ignored. The only exception to this rule is that twins that have been forced to stay homogeneous (using the GOHOMO routine, as described in Section 6) will then obey subsequent calls to CHGSUC. Subsequent calls to GOSLIP erase any such GOHOMO calls.

4.7.14.3 Twinned Paths: Behavior

This section endeavors to give the user some feel for how twinned paths behave compared to traditional (single, homogeneous) paths. This discussion includes transitions between homogeneous and slip flow modes, descriptions of how each flow regime differs, and details of new time step limits for twinned paths.

Single Phase Limits: Homogeneous Mode—As described above, twinned paths may exist in either one of two modes: (1) the slip flow mode, where both paths are active, the primary carries liquid, and the secondary carries vapor; and (2) the homogeneous mode, where the primary path acts like a normal path and the secondary path is largely ignored. While the user may purposely choose to avoid the slip flow mode (as described in Section 6), the decision to change modes and the corresponding conversion are handled automatically.

The homogeneous mode is intended to cover the extremes of single phase liquid and single phase vapor, but is actually encountered at very low and very high void fractions in two–phase flow. Hysteresis is used for stability. For example, twinned paths will transition from slip flow mode to homogeneous mode when the void fraction of the upstream lump is greater than 0.999999, but will remain in the homogeneous mode until the void fraction falls below 0.9999. At the liquid end, the limits of the hysteresis cycle are void fractions of 0.001 and 0.01. (Analogies with the perfect mixing vs. separated modes in the NONEQ0/1 routines should be evident.) If the flowrate in either path reverses and, in doing so, cannot extract the necessary phase, the homogeneous mode will result. In other extreme cases such as zero flow or 100% liquid adjacent to 100% vapor, the homogeneous mode will result.

When a pair of paths transitions into the homogeneous mode, the flowrate in both pairs is summed to produce a single flowrate, and the primary path is returned to a STAT=NORM phase suction state. The flowrate in the secondary path is set to zero, which can be used in logic to distinguish between the two modes.

When the upstream lump of a primary path enters the two–phase region sufficiently to enable slip flow modeling, the secondary path is reactivated and the flowrate in the primary path is redistributed between the two according to the current suction quality. At that time, the phase suction of the primary path is reset to STAT=DLS and the phase suction of the primary path is reset to STAT=DVS. As always in the slip mode, all shared parameters (e.g., DH, TLEN) are copied from the primary path into the corresponding locations for the secondary path.

Interactions with ACCEL—In addition to affecting the determination of the flow regime, the magnitude and direction of the ACCEL vector affect twinned paths by applying a body force on each phase in proportion to its density. In homogeneous flow, the ACCEL vector acts like a superimposed pressure force whose strength is proportional to the bulk fluid density. In slip flow, each path will feel a different force and react differently. For example, applying gravity to a vertical duct can cause vapor to rise at the same time liquid is falling: countercurrent flow might result (see Sample Problem G). This force is always applied according to the void fraction of the uphill lump for stability, independent of flow direction. Junctions are generally incompatible with perpetual countercurrent flow.

Flow Regimes, Interface Drag, and Wall Friction Distribution—When IPDC=6, one of four flow regimes will be attributed to twinned paths: two dispersed (bubbly and slug) and two separated (annular and stratified). These regimes and the pressure drops associated with them are described in Section 4.7.13. When paths are twinned, the flow regime also determines the interface drag and the apportionment of wall friction to each phase. These results in different behavior in each regime.

Despite the value of UPF, in the slip flow mode only the current upstream lump affects the flow regime and hence the subsequent wall friction and interface drag calculations. In other words, twinned paths behave as if UPF=1.0 in the slip flow mode, but then resume normal UPF usage when in the homogeneous mode. The only exception is countercurrent flow, where both endpoint lumps are used in a weighted average based on the flowrate of each phase.

Unlike homogeneous flow, there *is* a difference in behavior between bubbly and slug flows in twinned paths even without an acceleration gradient. One difference is in the interface drag calculation (FD). In bubbly flow, the average size of the bubbles is estimated on the basis of the slip ratio, void fraction, and surface tension, and then the drag of the bubbles through liquid is estimated. The result is a drag force that is a strong function of the slip ratio, tending to quickly 'homogenize' the flow and minimize slip. Because of this homogeneity, the wall friction (FC and FPOW) is attributed to both liquid and vapor phases in proportion to the friction that each phase would experience if the other were absent.

In slug flow, a correlation based on void fraction is used, which again estimates the drag of long bubbles in the liquid. In the limits of very high void fractions (e.g., vertical upflow where annular is prohibited due to flooding), a check is made to ensure that the drag force is at least as strong as it would be in separated flow. All wall friction is applied to the liquid phase and none to the vapor phase (which 'feels' the wall only by the interfacial shear).

In annular flow, the Lockhart–Martinelli correlation is generalized into an interfacial drag correlation whose limits are set by $f_{gl}/f_{gw} = 1$ on the low end (meaning a smooth interface), and a variation of Wallis' classic $f_{gl}/f_{gw} = 1 + 75(1-\alpha)$ corrected for variable density ratios (ρ_l/ρ_v) on the high end (meaning a very rough interface). The result of this treatment is that steady–state pressure gradients will approximately correspond to the Lockhart–Martinelli predictions that are implied by setting IPDC=2 for homogeneous paths. As with slug flow, all wall friction is applied to the liquid phase, and none to the vapor.

In the stratified regime, wall friction is apportioned according the phase velocities as well as the proportion of perimeter wetted by each phase (assuming a circular duct). The interface drag is calculated using the width of the interface (again assuming a circular duct), a Reynolds number based on the differential velocity, and a roughness estimated on the basis of the Froude number. The limits on this drag are $f_{gl}/f_{gw} = 1$ and $f_{gl}/f_{gw} = 10$.

The less the degree of slip (with homogeneous corresponding to zero slip), the greater the void fraction. Unfortunately, the greater the void fraction, the greater the tendency to be in a regime that allows greater slip. This would lead to instabilities were it not for the sizeable hysteresis employed in the flow regime determination logic. For example, as annular flow transitions to slug flow, the void fraction decreases as liquid is swept up by the faster moving vapor. However, as the void fraction diminishes, the transition back to annular becomes more likely, setting up the potential for an endless loop.

A similar problem exists at the liquid end of the homogeneous transition, where bubbly or slug flow becomes homogeneous below a void fraction of 0.001, and the subsequent elimination of slip drives the void fraction back up. Hence, a hysteresis band is used for that transition as well: homogeneous flow is allowed to persist up to a void fraction of 0.01.

Time Steps—Twinned tubes are subject to the same time step constraints as normal tubes, except that these limits are applied to each tube of the pair. Thus, the flowrate in *any* tube is kept from changing by a ratio of more than DTTUBF (or RSTUBF) during one time step.

Twinned tubes also obey an extra limit that does not apply to homogeneous tubes: the "SLIP FLOW CHANGE LIMIT." This is the change limit placed on the *difference* in velocity between the primary and secondary tubes. In other words, this limit makes sure that the slip ratio does not change by too much (again, DTTUBF or RSTUBF) per time step. Otherwise, the terms unique to slip flow (e.g., linearized drag, wall friction distribution, etc.) would accrue excessive error per solution step. Unfortunately, this limit is encountered frequently. The user can only avoid this limit by (1) avoiding twinned tubes, or (2) increasing DTTUBF/RSTUBF (which must be less than 1.0).

While STUBE connectors cannot directly restrict the time step, they have an indirect effect because limits are placed on the flowrate changes allowed in and out of adjacent tanks. For twinned STUBE connectors flowing out of two–phase tanks, the message "FR CHG LIM IN TWIN PATH" may appear if the time step is limited in order to indirectly limit flowrate changes that would be caused by tank void fraction changes.

4.7.15 Alternative Fluid Descriptions

FLUINT submodels require the complete description of the working fluid's properties. This description must include transport properties such as conductivity and viscosity, and *self-consistent* thermodynamic properties such as temperatures, pressures, and densities. As an example of self-consistency, if a temperature is calculated given specific values of pressure and enthalpy, $T_o = T(P_i, H_i)$, then the input value of pressure should result if it were calculated given the original enthalpy and the newly calculated temperature value: $P_i = P(T_o, H_i)$. This requirement for self-consistency does not preclude simplified or idealized descriptions.

Extensive data is required to completely and consistently describe a two-phase working fluid from subcooled liquid, through and around the saturation dome, to superheated and possibly supercritical vapor. For this reason, the descriptions of 20 two-phase refrigerants are available in prestored form (see Appendix C); these fluids are called the *standard library fluids*. Of course, this limited selection does not fulfill all modeling needs. Also, the cost of self-consistency for full functional two-phase descriptions is high; all properties must be calculated as if a liquid could boil and a gas could condense, and potentially restrictive limits on temperatures and pressures are necessary. Therefore, the user may also define the properties of alternate working fluids: *user-defined fluids*. Four types of alternate fluid types are available:

1) A perfect gas (that cannot condense). Both the specific heat and transport properties may vary with temperature. In other words, the gas does not have to be calorically perfect.

2) A simple liquid (that cannot boil). The liquid is assumed to be incompressible, but density as well as transport properties can vary with temperature.

3) A simplified description of a two-phase fluid. This option is somewhat like a combination of the above two fluids, with saturation dome and two-phase properties also required. This option is intended to allow a limited description of a two-phase fluid to be quickly added as an alternate fluid using commonly available properties.

4) An advanced (complete) description of a two-phase fluid. The user may replace some of all of the standard library property routines with alternates; in other words, the user may input a functional or tabular description of a fluid over a very wide range of properties, *including supercritical if desired.*

The ability to define new fluids significantly expands the usefulness of the program by allowing simulation of more fluids over a wider range of temperatures and pressures. Furthermore, the program runs an order of magnitude faster using a simplified description of a standard library fluid (such as liquid water). Tabular descriptions, correctly done, also exhibit both fast execution and a wide domain—the two are normally mutually exclusive.

The remainder of this section describes the formulations and conventions used in the fluid property calculations. Refer to Section 3.13 for input formats and a more detailed description of two-phase fluids, Appendix C for property ranges, and Appendix D for unit system conventions. Refer to Section 7.3 for a discussion of RAPPR, which generates simplified descriptions of the standard library fluids for faster execution.

Unfortunately, generating tabular fluid descriptions is not trivial. Rather than attempt to document the algorithms and procedures required to duplicate such descriptions, the user is advised that tabular descriptions of water, ammonia, hydrogen, nitrogen, oxygen, ethane, and argon are available and should be used as templates for other fluids.

Perfect Gases—The basic relationship for a perfect gas is:

$$P = \rho R T$$

where:

- P ..... the absolute pressure
- ρ ..... the density
- R ..... the gas constant, $\mathfrak{R}/MW$, where $\mathfrak{R}$ is the universal gas constant and MW is the molecular weight of the gas
- T ..... the absolute temperature

The specific enthalpy, h, and internal energy, u, are defined by:

$$h = \int_{T_{ref}}^{T} C_p(\tau)\, d\tau$$

$$u = h - P/\rho = h - RT$$

where:

- T_{ref} ..... an arbitrary reference temperature, chosen as absolute zero
- C_p ..... the specific heat at constant pressure

Because of the chosen reference temperature, enthalpies will always have positive values.

Incompressible Liquids—Because of the assumption of incompressibility, pressure becomes unimportant in the property calculations for these liquids. (Actually, the assumption of compressibility can be added with any working fluid by using compliant tank walls.) Because there is little difference between the specific heats at constant pressure or constant volume for a liquid, the enthalpies for liquids are calculated as follows:

$$u = \int_{T_{ref}}^{T} C_p(\tau)\, d\tau \qquad (C_p \approx C_v)$$

$$h = u + P/\rho$$

Pressures are allowed to range from -1.0E10 to 1.0E10. Enthalpies will almost always be positive values, except for extremely large negative values of pressure. Note that a model using a simple liquid as a working fluid will require almost an order of magnitude less CPU time than one using a standard library fluid that never boils (refer to Section 7.3).

Simplified Two-Phase Fluids—For the P-v-T surface of the gas phase, the van der Waal's equation (a slight improvement over the perfect gas assumption) is used:

$$P = \frac{RT}{v-b} - \frac{a}{v^2}$$

Although some handbooks list constants a and b, they can be calculated given the critical point for the fluid. This is the best general P-v-T relation that can be calculated on the basis of simple user-provided properties. However, the approximation is not good near or above the critical pressure, and might be inadequate for liquid metals and highly polar molecules.

For the saturation pressure/temperature relation, the following general equation is used:

$$\ln P = A + B/T + C\ln T + DT + ET^2$$

This relationship is adequate for most fluids over a limited range. The five constants in the equation are calculated by the program on the basis of the critical point, and two values each of saturation pressure and heat of vaporization at given saturation temperatures.

Enthalpies, internal energies, and entropies are found by closed form integration of the P-v-T surface from zero density, and numerical integration of the C_v° curve (that is, the curve of C_v versus temperature at the limit as pressure goes to zero) from the reference temperature, which is absolute zero:

$$u = \int_{T_{ref}}^{T} C_v^\circ(\tau)\, d\tau + \int_0^{\varrho} \left[P - T\left(\frac{\partial P}{\partial T}\right)_{\varrho} \right] \frac{dr}{r^2}$$

$$h = u + P/\varrho$$

These equations can be used to calculate saturated vapor properties. Across the saturation dome, the Clapeyron equation ($h_{fg} = Tv_{fg}(dP/dT)_{sat}$) is used to find the heat of vaporization. The dP/dT derivative is found from the saturation pressure curve. Thus, saturated liquid properties can be calculated. For subcooled states, a $(P-P_{sat})/\varrho$ term is added to the enthalpies. Because of these methods, all properties are thermodynamically consistent, and are all referenced ultimately from a low pressure gaseous state.

4.7.16 Modeling Tips for Efficient Use

FLUINT efficiency is much more dependent on good modeling practices than is SINDA. Unexpected results and high solution costs are more likely, but these can be controlled by carefully approaching the modeling problem. This section presents some useful tips for getting the most out of a finite computer budget.

Spatial resolution—Solution costs for FLUINT increase with the almost the square of the number of tanks and junctions (approximately), but plena are almost free. These facts should be weighed into any modeling decision. In other words, the user should use the minimum number of tanks and junctions that will satisfy the modeling requirements. Unjustified high resolution can be costly. Fewer, larger tanks should be a goal.

Also, plena should be used not only for open systems but also wherever the thermodynamic state of a lump does not change appreciably. For example, a single fluid system can be subdivided into separate, smaller submodels using plena if enough locations are found that can be approximated by a plenum. If a 100 lump submodel were subdivided into two 50 lump submodels, the resulting two submodel system could be solved in about half the time of the former single submodel system. Also, consider using a plenum not only to set a reference pressure but also as a perfect heat exchanger (see also heater junctions and the HTRLMP routine.)

One easy way to eliminate unnecessary lumps is to combine paths. There is usually little need to resolve pressure gradients along most lines. For incompressible flow, all fittings along a line can usually be combined algebraically into a single LOSS element or valve model. Furthermore, LOSS elements can be eliminated by applying an FK to a nearby tube or STUBE connector. *When creating detailed models that are faithful to the system schematic, the user should remember that he may be implicitly asking the program to split analytic hairs.*

Two cautions against overly coarse models are in order. First, the heat transfer rates will be underestimated in single–phase flows if the model is too coarsely discretized and HTNS or HTUS ties are not used. Second, a junction (or FASTIC tank) downstream of a high impedance tube or STUBE connector (with UPF less than 1.0) may experience an artificially high pressure drop if the flow is compressible, possibly to the point of causing property warning messages. Both of these problems are due to violations of lumped parameter modeling practices: inadequate discretization to resolve an important temperature gradient in the first case, and inadequate discretization to resolve an important pressure (and therefore density) gradient in the second case.

Temporal resolution—Solution costs for FLUINT also increase dramatically as the resolution of time-dependent changes is increased. Whereas spatial resolution refers to the *number* of elements used to model a system, temporal resolution refers to the *type* of element. A network built entirely from tanks and tubes represents the highest possible temporal resolution (and is able to resolve transient temperature and pressure histories that occur over intervals in the range of microseconds to minutes), whereas a network built entirely from junctions, plena, and connectors represents the lowest possible temporal resolution (i.e., the network responds instantaneously to changes in parameters or boundary conditions). As a reminder, however, no network can be built entirely from junctions, and diabatic junction loops can cause problems. *For most thermal-structural analyses with a characteristic time on the order of minutes to hours, an instantaneous (junction, plenum, connector) model will suffice.* If specific events inside of the fluid system need be resolved (such as control transients), the user may need to resort to a more temporally detailed model.

With respect to single-phase liquid systems, if fluid lag times are unimportant and energy can be assumed to propagate instantaneously (at least, compared to time scales of interest) around the network or any portion of the network, use junctions instead of tanks. However, the cost of handling a rigid single-phase tank is not significantly higher than that of a junction, and junction heat transfer calculations are not easily modified. A compliant tank is somewhat more expensive to analyze than a rigid tank, but has numerous advantages. Conversely, the cost of handling two-phase tanks *is* significantly higher than that of junctions—avoid small two-phase tanks unless energy lag times and/or liquid inventory changes are important.

In general, use STUBE connectors instead of tubes if flowrates can change discontinuously or are expected to have zero flow. Tubes should not always be avoided, however. They are actually cheaper to handle than STUBE connectors per solution step, and although they can directly cause a smaller time step, connectors can also restrict the time step indirectly due to their effect on adjacent tanks. Tubes also allow easier custom tailoring of friction and acceleration calculations (AC, FC, FPOW, etc.) in FLOGIC 1. As a rule of thumb, *use tubes with tanks and STUBE connectors with junctions for duct models, especially for gaseous and two-phase flows.*

Carefully selected initial conditions—FLUINT cannot know whether the initial conditions given in the preprocessor were intentionally chosen for a transient analysis or were simply guesses at a steady-state solution. The program may therefore spend a lot of time trying to achieve a steady-state solution from careless initial conditions. Although it is not mandatory, it is well worth the user's time to make sure that "guessed" initial conditions are at least self consistent. This includes making sure that flowrates around loops and into lumps sum to zero, and that zero flowrates are only input where no flow is expected. Also, once a useful steady state has been achieved, it should be saved for future use with RESAVE or SAVPAR calls in the user logic blocks. The use of FASTIC can greatly reduce the time spent finding valid initial conditions, but cannot always resolve nonsensical inputs. Also, hysteresis is not uncommon; in some models more than one equally valid steady-state solution can result given different initial conditions.

Excursions beyond the property routine limits are not unusual during steady-state runs, and only indicate problems if they persist or preclude convergence. The routines CHGLMP, HLDLMP, HTRLMP, RELLMP, and SPRIME (for capillary devices) can all be used to control or guide steady-state runs. Also, many fluid systems simply have no steady-state: unlike thermal systems, they may oscillate indefinitely even with fixed boundary conditions.

Preliminary runs—Most of the problems and unknowns discussed above can be resolved by making preliminary runs with coarse models (less spatial or temporal resolution) or with models of subnetworks (broken down using plena as boundary conditions). This will provide both good initial conditions as well as help resolve modeling choices such as how many segments to use for a heat exchanger.

Conserve volume—Only tanks have definite volume. However, as mentioned before, tanks should be used only where needed. Junctions can provide much of the same data without slowing down transient analyses. However, the user should never "throw away" volume when using a junction. Rather, it should be added into the volumes of adjacent tanks. For example, tanks are inconvenient to use when modeling a tee, but the volume within the tee (and any connecting lines) should be attributed to nearby tanks through which the flow moves.

Include accumulators or reservoirs in closed system—In real systems, accumulators serve several purposes such as dampening pressure fluctuations, absorbing volume changes caused by thermal expansions/contractions and by vapor or gas generation, and making up for fluid lost to leaks. Almost all real fluid systems have some type of accumulator, and they should not be neglected in the corresponding fluid model. Both tanks and plena may be used as accumulator models. Plena should be used unless the performance of the accumulator itself is important. (Any junction or tank may be dynamically placed into a boundary state and returned to its normal state using the HLDLMP and RELLMP routines.) In general, the presence of a plenum in a closed system will smooth system operation and could reduce the number of solution intervals needed for both transient and steady-state analyses. In fact, *if there are any rigid tanks used in a model of a completely hard-filled (single-phase liquid) system, a plenum must be present.* Otherwise, any thermal expansion or contraction of the fluid would cause extreme pressure fluctuations, possibly exceeding the range of available properties. If tanks are used as accumulators, the VDOT, COMP, and QDOT values of the tank can be manipulated as necessary to simulate the accumulator response, and many simulation routines are available such as NONEQ0/1, BELACC, etc.

Smoothing sensitive systems—Many systems are physically realistic but very delicate due to numeric approximations. The user can help make models more rugged by making them more tolerant—by removing some of the more inflexible assumptions.

Placing LOSS elements in parallel with MFRSET devices is an example of smoothing. Because meeting the requirements of an unforgiving (GK=0.0) MFRSET connector can be occasionally difficult for the rest of a system, a good practice is to place a LOSS element with a large FK factor in parallel with the MFRSET connector to act as a bypass to reduce pressure surges. The FK factor should be sized to allow a small percentage of the total flow to bypass the MFRSET device. This sizing can be done within the FLOGIC 0 block during a steady state run. This is but one example of the usefulness of multiple, parallel paths to model a more complex device.

Another example of smoothing is using compliances in otherwise rigid tanks to allow them to respond to flow changes, as discussed previously. In fact, noncondensible gases are sometimes introduced purposely into liquid systems to achieve the physical equivalent of this numeric method. Refer also to the COMPLQ routine in Section 6.9.6.7.

Two-phase lines and heat exchangers—Junctions should be considered instead of tanks in heat exchanger and pipeline models where two-phase flow is expected. *Small, two-phase or vapor (soft) tanks can significantly slow the solution.* However, if liquid inventory changes and/or lag times are important, tanks can be used. In this case, *it is recommended that tubes be used instead of STUBE connectors when modeling two-phase ducts with tanks.* Otherwise, the instantaneous nature of connectors combined with the very time-dependent nature of soft tanks and the nonlinearities of two-phase correlations can lead to instability. Consider setting DTTUBF=0.5 for models where tanks are required, and tubes are used for stability reasons. Otherwise, the tubes will tend to dominate the time step limits. Even with junctions, STUBEs may allow the flowrate to change too quickly, especially in condensers where deceleration (i.e., AC and FR are the same sign) can be dominant. Also, as mentioned in 4.7.5, center-lumped line and heat exchanger segments are much more desirable for two-phase ducts.

Constant heat flux vs. constant wall temperature—Text book heat transfer problems are generally broken down into two ideal types: constant heat flux and constant wall temperature. Of course, most real systems are somewhere in between these two extremes but can be modeled as one or the other. With FLUINT, constant heat flux cases (such as electrical dissipative heating) should be modeled using the QDOT parameter on a single lump if no thermal model is used for heat acquisition. Otherwise, constant wall temperature cases (usually the case during heat rejection) should be handled with the minimum number of heat exchanger segments that will satisfy the modeling accuracy requirements. (HTUS and HTNS ties tolerate very coarse resolution for constant wall temperatures if the flow is single–phase.) These are rules-of-thumb only. As with real systems, *constant heat flux is dangerous if the flowrate through the heated or cooled section drops off unexpectedly.*

Ideal control systems—Control systems should be idealized where possible. Fluid submodels are much more sensitive to poor control logic in user logic blocks than are thermal submodels. Temperature (or quality) and pressure control in a line can be achieved simultaneously or independently, as shown in Figure 4–22. For both temperature and pressure control, a plenum can be inserted in series with the line to be controlled. To control only pressure without affecting the line temperature or quality, connect a plenum to a line using an STUBE connector. The flowrate in this connector will be negligibly small, especially for single-phase systems. To control temperature but not pressure (i.e., an ideal heat exchanger or mixing point), a heater junction may be used. A heater junction holds a constant enthalpy, adjusting the QDOT as needed (see routines HTRLMP and RELLMP in Section 6).

Flowrate control can be achieved using MFRSET connectors, which will maintain the given flowrate independent of flow conditions or pressure gradients. Also useful are VFRSET and VPUMP devices. More complicated controls are possible with CTLVLV devices, which can be used to open up and shut off whole subnetworks. Pressure control valves may be simulated by manipulating the FK of any LOSS-type connector, but the user should also consider the use of the HC factor in an STUBE connector for improved stability and perhaps simpler logic.

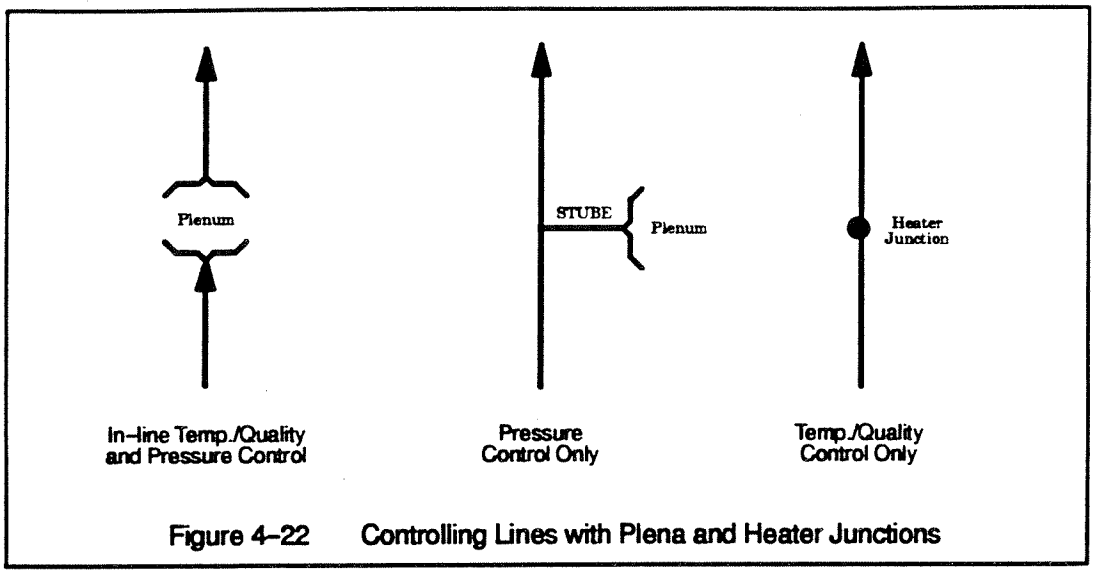


Figure 4–22 Controlling Lines with Plena and Heater Junctions

Feedback control system modeling—One of the capabilities that makes FLUINT unique is its ability to handle a wide variety of feedback control systems via the user logic blocks (usually FLOGIC 0), analogous to the way that SINDA can be used to simulate thermostatic control logic (usually in VARIABLES 1). Among the parameters that can be varied include pump outputs or shaft speeds, valve or loss element resistance values or open/close status, and heat and volume rates. These can be varied according to any desired logic using any available system variable.

However, there are several cautions that must be observed when modeling flow control systems. First, the control system may be *physically* unstable, causing oscillations, nonconvergence, or property range excursions. Note many real control systems produce oscillations in the system when working properly. Second, the control system may be *numerically* unstable—the real control system would work properly but its representation within FLUINT is inadequate. While one of the principal uses of FLUINT is to evaluate such control systems, the modeling of the action/reaction logic *is the user's responsibility.*

Some of the key points to remember about modeling such feedback control systems within FLUINT are:

1) The input or sensed system variable should not be able to change discontinuously (such as a connector flowrate or a junction state). This would cause steady-state solutions to be troublesome, especially with FASTIC, and might require high temporal resolution to resolve. If the input variable *can* change discontinuously, it should be fairly independent of the control system reaction (which would be unusual for fluid systems where everything usually affects everything else).

2) The output or reaction of the control system should not be abrupt—it should occur gradually giving the system adequate time to respond to the changes. If a control system is causing problems, the output can be artificially damped. For instance, logic can be added to avoid binary oscillations, or to limit the change in the response each pass through the logic, perhaps by using a running average. While the future

time step can only be guessed, the last time step is available in the form of the control constant DTIMUF, and is frequently used for PID control simulations. This constant is initially zero.

3) Use idealized control systems wherever possible. Instead of faithfully tracking transient changes, the output of the control system may be controllable directly. For example, use a plenum to control a particular pressure rather than a detailed model of a regulator. Also, use an MFRSET connector to maintain a given flowrate in a bypass branch rather than a detailed model of a three-branch valve. Use the HC factor on an STUBE to model a pressure regulator rather than attempting to dynamically size an orifice or valve coefficient.

Updating Factors for Instant Elements in Logic Blocks—Similar to the concerns voiced above, many FLUINT parameters can cause immediate and perhaps severe changes in network behavior, perhaps to the point of invalidating the assumptions made when updating the parameter. For example, updating the UA on an HTU tie to a junction can instantly (at least, in the next solution step) change the state of that junction to the point where the rationale used to choose that particular UA value (perhaps the junction quality) is no longer valid. The following is a list of such instant variables, which apply to all connectors and junctions in any solution routine, as well as to tubes and tanks in FASTIC:

QDOT	on junctions and FASTIC tanks
UA, DUPL, DUPN	on ties to junctions, FASTIC tanks, and arithmetic nodes
DUPI, DUPN, (STAT)	connectors and FASTIC tubes
FC, FPOW AC, HC	STUBE connectors and FASTIC tubes
DH, AF, TLEN, FK	STUBE connectors and FASTIC tubes
UPF, IPDC, WRF	STUBE connectors and FASTIC tubes
FD, FG	STUBE connectors and FASTIC tubes
FK (FKB)	LOSS family of connectors (includes valves)
SMFR	MFRSET connectors
SVFR	VFRSET connectors
SPD	VPUMP connectors
RC, CFC, XVH, XVL	CAPIL connectors, CAPPMP NULLs
GK, HK, EI, EJ, DK	NULL connectors

4.7.17 Output Options

This section describes the output routines available to users. These routines are summarized in Table 4–6. See Section 6.9.5 for more details. As a minimum, users should consider LMPTAB and PTHTAB in their OUTPUT CALLS blocks.

Table 4–6 FLUINT Output Routine Summary

ROUTINE	ACTION
LMPTAB	Tabulates lump parameters
LMXTAB	Tabulates additional lump parameters
TIETAB	Tabulates tie parameters
PTHTAB	Tabulates path parameters
TWNTAB	Tabulates path parameters for twinned paths
TUBTAB	Tabulates tube and STUBE connector parameters
LMPRNT	Prints lump thermodynamic state
QDPRNT	Prints lump (tank and junction) heat rates
TKPRNT	Prints tank masses, volumes, and volume rates
FRPRNT	Prints path flow rates (including path type and connector device model names)
TBPRNT	Prints tube DH, AF, TLEN, Reynolds number, TLEN/DH
STPRNT	Same as TBPRNT for STUBE connectors
FLOMAP	Prints network linkages and mass and energy flow
LMPMAP	Same as FLOMAP for one lump
PRPMAP	Prints mapping of working fluid properties

4.7.18 Execution and Control

Steady-state analyses of all active submodels (whether fluid or thermal) are performed by calling STDSTL or FASTIC. Similarly, transient analyses are performed by calling FORWRD or FWDBCK. Note that both FORWRD and FWDBCK use the same solution method for fluid submodels even though the methods are different for thermal submodels. In fact, STDSTL uses the almost same fluid solution method, but uses large artificial time steps to relax the network towards a time-independent solution.

FASTIC, which employs the same thermal method as STDSTL, uses an iterative approach for fluid models. FASTIC treats tanks like junctions and tubes like STUBE connectors. Because of this treatment, some options (such as tank volume factors) are either ignored or approximated. In general, FASTIC approaches steady-state solutions very quickly but may not converge easily, while STDSTL approaches solutions slowly but almost always converges with all options active. A good practice is to call FASTIC and then STDSTL when fluid submodels are active. If the FASTIC call converges, then a subsequent STDSTL call will typically take two to four more iterations to confirm that solution.

Three user logic blocks are available for each fluid submodel: FLOGIC 0, FLOGIC 1, and FLOGIC 2. These are similar to VARIABLES 0, 1 and 2 but do not exactly correspond to them. FLOGIC 0, the primary location for most user logic, is called before each solution interval (meaning one time step for transient analyses or one relaxation solution for steady-state analyses). FLOGIC 1 is called after FLUINT has automatically calculated all required element descriptors (such as any QDOT, connector GK and HK, tube FC, etc.) to fulfill any requested or implied calculations. This call (FLOGIC 1) occurs before the time step (or steady-state relaxation step) is calculated, and is the last chance to alter the parameters before the solution begins. FLOGIC 2 is called after the solution has been performed and the network variables have been updated. In other words, the FLOGIC 0/1/2 sequence can be thought of as initialize/tailor/wrap-up. The parameters available to the user in these blocks are described in Section 3.12.2.

In order to allow complete access to all model parameters (for heat transfer or control simulation calculations), both fluid and thermal parameters are available in *any* logic block (fluid or thermal). In general, however, *the user should only alter fluid parameters in FLOGIC blocks,* and similarly should only alter thermal parameters in VARIABLES blocks.

4.7.18.1 Steady-State Convergence

As noted above, the fluid network methodology in STDSTL is very similar to the transient methods used in FORWRD and FWDBCK. In other words, the program takes artificial time steps, integrating the solution towards a time-independent state. In FASTIC, the time-independent equations are solved iteratively until no significant changes are noticed. Unlike all other solution routines, FASTIC normally* solves the hydrodynamic equations separately from the energy equations during each solution step. Both STDSTL and FASTIC, however, use the same methods to guide and detect convergence.

* When twinned paths are used, or when EI or EJ is nonzero for some path, then FASTIC must solve both energy and hydrodynamic equations simultaneously.

There are two types of convergence criteria: the relative change in a certain parameter between iterations, and the relative error in the energy flow through either a lump or the entire submodel. "Relative" means that the error is expressed as a unitless fraction of the relevant parameter.

The following parameters are subject to the first type of check:

1) Lump TLs (absolute value)

2) Lump DLs

3) Lump PLs (absolute value, skipped for 9000 fluids)

4) Path FRs (unless the flowrate is judged insignificant)

5) Path pressure differentials (unless the flowrate is judged insignificant)

6) Path enthalpy differentials (for twinned paths, or paths with nonzero EI or EJ)

To be converged, the relative changes in all of the above parameters must be less than RERRF, which has a default value of 0.01 (1%). Typically, the first three criteria are met quickly, while the last two take more iterations to satisfy. The program loops through the elements performing these checks, and the first lump or path to fail this check is listed in the "CONVERGENCE STATUS" message that precedes most output routines. This should not be interpreted as meaning that only this element failed the test, or that its failed by the largest margin—it was simply the first. Furthermore, lumps are checked before paths. Thus, a model that lists a lump density change as the reason for nonconvergence is not as close to a final solution as is one that lists a path pressure differential as the reason.

Upon successfully satisfying the relative change criteria, the program will attempt to satisfy the energy balance criteria, as governed by REBALF. If REBALF is zero, this check is omitted. If positive, the energy flow for the entire fluid submodel is checked by summing QDOTs into nonplenum lumps and energy flows in and out of plena. In this context, "plena" includes lumps that have been held with a call to HLDLMP. The submodel may be considered converged if the energy flows balance within REBALF, or if they do not balance but are stationary and show no progress toward balancing. This latter case is normally caused by duplication options, which can artificially magnify one the energy flows if a virtual subnetwork is not adiabatic. In such cases, the convergence message will be accompanied by a warning that "ENERGY FLOWS ARE STABLE BUT UNBALANCED." (Costly methods such as matrix inversions would be required to calculate how many times a lump *really* appears in the master model from having been duplicated.)

If REBALF is negative, then the energy balance check is performed on a lump-by-lump basis, with |REBALF| being compared with the largest imbalance of any nonplenum lump. Unlike its analog EBALNA, which has units of power, negative REBALF is interpreted as a unitless (fractional) error in the energy rates for each lump.

The routine LMPTAB can be used to help detect problem areas in convergence. The last two columns for each lump list the current net rates of mass and energy increase for all lumps, including plena. Upon convergence, these rates should be very small for all nonplenum lumps. Lumps with large mass and energy rates indicate possible problem areas, especially in STDSTL which uses matrix methods for simultaneous solutions. In FASTIC, the energy rate for a lump

may simply be off because it has not been updated recently during the iterative solution. (The FASTIC energy balance uses a mixing algorithm to vary the order in which lump energy iterations proceed.)

Note that work terms are ignored in the energy balance. Work on or by the system from VDOT (volume changes) are ignored because VDOT must eventually be zero to achieve steady-state (although the user may manipulate VDOT values during the relaxation process). Work by pumps, body forces, and frictional losses are also neglected in the energy balance because they are neglected in the model itself.

Actually, the real solution procedure is considerably more complicated than it would appear from the above discussion. To help avoid property range errors and to absorb very rough initial conditions, the steady state routines start very cautiously in a sequence called "PRELIMINARY ITERATIONS" in the output message. During these iterations, the network is pushed very gently towards a solution, and the rate of convergence is compared with 10 times the values of RERRF and REBALF (i.e., 10*0.01 = 0.1, or 10% by default). Once the network satisfies these very coarse criteria, the program starts to push harder and harder, until the network is able to satisfy the real convergence criteria and the program is not being artificially gentle. (The variable degree of "pushing" or "gentleness" is achieved by scaling the driving terms, namely path HK values, in the hydrodynamic solution from their full values.) Thus, if NLOOPS is exceeded and the convergence status says "PRELIMINARY ITERATIONS: 10*RERRF NOT SATISFIED," the user should be advised that the network is very far from being converged. On the other hand, if the solution is pushing harder and the network is responding well, a message such as "CONVERGENCE PENDING" can result.

As a result of the above methods, it takes at least two iterations to confirm convergence of a model that has already been solved. If thermal submodels are active, four iterations are required as a minimum.

4.7.18.2 Fluid Submodel Time Steps

In transient routines as well as in STDSTL, FLUINT calculates its own maximum allowable time step. In STDSTL, this is the step size used, and it may be different in each fluid submodel. In other routines, this time step is then compared with the step required by other submodels, and the step required for output intervals, etc. The minimum of all such limits is used as the time step in transient routines. Control constants RSSIZF, DTSIZF, RSTUBF, and DTTUBF are used to help determine the allowable time step per submodel. Constants DTMAXF, RSMAXF, OUTPTF, etc. are used to impose other limits on the time step. DTIMUF contains the result of the time step calculation for each submodel, and the output header describes its causes.

This section focuses on the first process: determining an allowable step size based on current network status, expected changes, and previous behavior. In the following discussion, "RSSIZF" may be substituted for "DTSIZF" when referring to STDSTL. DTSIZF is used as the generic term. Similarly, reference to DTTUBF also implies RSTUBF.

Background: Integration Strategies—FLUINT uses a first–order implicit time step integration that is performed in parallel with whatever method is used to integrate the thermal networks. Heat rates between thermal and fluid models are held constant to conserve energy. If all property domains and derivatives, friction coefficients, heat rates, etc. truly remain constant over the time interval, then the solution is fully implicit and an arbitrarily large time step can be taken. Since these parameters in fact often vary, a best estimate is made of the time step that can be made without excessive changes in such parameters.

Extensive logic is employed to estimate this time step and to check predicted changes against the previous step. While this feedback method successfully avoids time steps that are too small (from the mathematical standpoint if not from the user's standpoint), the only way to be absolutely sure that this estimated time step is not too big is to proceed to integrate the equations and solve for the next network state. If unforeseen changes in operating regimes, boundary conditions, or other parameters are excessive, then at best excessive error will have been generated. At worst, the solution will fail or find a spurious answer such as negative masses in control volumes, or excursions beyond fluid property limits. In such cases, the program is able to back up and try again with a smaller time step.

In FLUINT, the selected strategy is to spend about 10% to 20% of the cost per solution to make a good and somewhat conservative estimate of the time step, and then back up and repeat a solution if need be. A strategy taken in other codes is to take a user–input time step, and then solve iteratively (typically on the order of ten iterations per time step, each about the cost of one FLUINT time step) for the final state. Instead of predicting time steps, the challenge becomes how to converge efficiently on a perhaps elusive final state.

Time Step Predictor—The FLUINT time step predictor faces two challenges. First, because of the implicit nature of the solution and the instant nature of many elements, time steps can only be estimated because anything can happen during the subsequent time step. Second, accuracy is crucial. If the time step is too big, severe errors can result and the time step will have to be repeated; if it is too small, the program can appear to halt with no output.

Unfortunately, there is no fluid equivalent of CSGMIN, and there is no maximum time step due to stability criteria (although tubes can exhibit a close analogy to such a mathematical limit). The maximum time step is therefore estimated such that the assumptions implied in the current solution step are substantially true throughout the next time interval.

This estimation takes the form of imposing two kinds of limits: *percent change limits* and *transition limits*. Percent change limits keep flowrates, densities, and heat rates from varying excessively during the time interval. Transition limits, on the other hand, keep major assumptions from being violated. For example, when a tank moves from all-liquid to two-phase or the reverse, almost all assumptions, governing equations, and property derivatives change. To step too far past such a transition would result in lower accuracy at the very least, and execution errors at the worst. To try to stop exactly at the transition point is too demanding and can result in perpetually decreasing time steps. Therefore, in such cases the program attempts to step just slightly past the transition.

The key to the time step prediction is the estimation of anticipated changes. This estimate must be as accurate as possible without requiring the full network solution. Some readers may be surprised to find that up to 20% of the CPU costs are devoted strictly to time step estimation, and not to the solution itself. Because of the criticality of the selected time step, this is money well spent.

The methods involved in this estimation are among the most complex methods in FLUINT, and will only be introduced here. Detailed knowledge is not required to use the code, but an overview is provided to assist the analyst in understanding program behavior. The time step prediction procedure is as follows.

First, allowable limits (both change limits and transition limits) are calculated based on current network status.

Second, a *local solution* is performed for each time-dependent element (i.e., tanks and tubes). This local solution assumes that adjoining network elements will not change over the next time interval. The collective results of these predictions are saved in memory for comparisons with subsequent network–level solutions during the next time step.

Third, the results of this local solution are corrected using previous network behavior—a feedback loop exists. This feedback correction is somewhat subtle and perhaps counterintuitive. If the local solution from the previous time step does not match with the local current solution, then an unforeseen change is presumed to have occurred which invalidates extrapolations of previous network behavior. Otherwise, if the previous and current local solutions agree well, then they are both neglected in favor of extrapolations of previous behavior. A degree of confidence is thus assigned to the local solution compared to extrapolations of previous behavior. One of the key purposes of this feedback control is to prevent *sticking*: continually small time steps with no appreciable progress made in the integration. Sticking would occasionally result if the program depended only upon local solutions to guide the time step prediction. Unfortunately, while this method helps prevent catastrophically large time steps, it cannot guarantee their complete avoidance.

Therefore, a fourth step is taken *after* prediction of the time step and indeed after the integration is complete: the results of the integration are reviewed to assure that the resulting answers did not violate the assumptions of the integration or the limits set by the user (e.g., DTSIZF, DTTUBF), thereby generating errors or instabilities. If the results of the integration are rejected, a prediction is made of the acceptable time step, the network state variables are reset to their previous values, and the integration is repeated with a smaller time step.

Because of the expense associated with repeating a time step, small errors are tolerated. Furthermore, the size of subsequent time steps may be reduced strictly to prevent an excessive percentage of repeated steps. If such a limit is being imposed, the status message will contain the words "EXCESSIVE BACKUP LIMIT." If the rate of backups continue to be high, a warning message will be produced.

Tube Limits—No tube may change flowrates by a fraction greater than DTTUBF (10% by default). The local solution is estimated by taking the allowable flowrate change divided by the current derivative of flowrate with respect to time—the end point lumps are assumed to not change. To avoid getting stuck on flows through side passages, this allowable flowrate change is not allowed to be smaller than a small percent of the submodel average flowrate. If the time step is limited by a tube, the status message lists that tube, and lists the reason as "FLOWRATE CHANGE LIMIT." For models where the stabilizing behavior of tubes is desired, but not the small time steps, consider setting DTTUBF=0.5.

For tubes for which AC*FR has become a large positive number (e.g., those at the inlet of condensers), an inherent stability limit exists which is flagged in output headers as "STABILITY LIMIT." This limit, while fortunately rare, can only be avoided by changes to the model to reduce the rate of deceleration (condensation, suction to side flow passages, rate of increase of flow area) or increase the frictional of head (K factor) losses within the tube. If such changes are inappropriate and the time step is unacceptably small, consider STUBEs instead.

For twinned tubes, the rate of change of the slip velocity (i.e., vapor velocity minus liquid velocity) is also constrained to changes no greater than DTTUBF. If this is the cause of the time step limit, the output headers will describe it as a "SLIP FLOW CHANGE LIMIT."

Tie Limits—The time step can be limited by heat transfer ties to diffusion nodes unless the ties are on single-phase junctions. Because the heat rates are assumed constant over the solution interval, this limit prevents the temperature difference from changing by an unacceptable amount. Because SINDA also assumes some responsibility for the time step of the tied node, this limit is relatively liberal.

The UA for the tie is summed into the SINDA CSGMIN calculation to appropriately burden the SINDA time step calculation. (Caution: the large effective UA values typical of HTUS and HTNS ties can cause reduced CSGMIN. If this is the case, consider tying to arithmetic nodes instead.) The local solution is calculated by estimating the derivative of the temperature difference with respect to time. Unlike tube and tank limits, no feedback control is currently applied to tie limits because (perhaps unlike the elements to which they connect) ties are strictly a time-independent element. If a tie limits the time step, the reason is listed as a "NODE TEMPERATURE CHANGE LIMIT," and the ID of the tie is listed.

Tank Limits—The limits for tanks are numerous, and vary according to the type of tank. The local solution assumes that adjacent lumps (and most adjacent flowrates) stay constant over the interval, and a reduced set of simultaneous equations is solved.

Transition limits include movement in and out of the saturation dome, and pressure gradient limits imposed by nearby capillary devices (whose behavior can change drastically when the pressure difference is near the capillary limit). Such limits are denoted "PHASE TRANSITION LIMIT" or "CAPILLARY TRANSITION LIMIT." Also, limits are imposed on the percent change in lump mass, density, volume, temperature (including limits due to any ties), energy (relative to the critical point), etc. In many of these limits, the default change limit may be less than DTSIZF, in which case the change limit actually used will be proportional to the value of DTSIZF. For example, the percent density change in a liquid tank is limited to 0.006*DTSIZF, which corresponds to 0.06% for the default value of DTSIZF. Accuracy would be questionable if the change were greater than this limit. Changes due to such effects are noted as "DENSITY CHANGE LIMIT," "VOLUME CHANGE LIMIT," etc. In tanks with XL=0.0 (all liquid), the temperatures and densities are directly linked, which can result in a "TEMP./DENSITY CHANGE LIMIT."

If pressure changes in a lump are likely to affect the flowrate in an adjacent connector so much that the flowrate could easily reverse, change flow regimes, or affect the tank itself, then the time step might be reduced even though connectors themselves are time-independent. This conservative measure avoid oscillations and potentially significant errors. When this limit is encountered, the reason is attributed to the tank, and is listed as "FR CHANGE LIMIT IN PATH

___." The time step is not restricted, however, if the connector flowrate is unstable since the instability is likely to persist even at diminishingly small time steps. Furthermore, time steps are never reduced by more than 50% due to this cause.

An analogous limit may be placed on the enthalpy change of a tank if twinned STUBEs flow out of it, in which case the message will read "FR CHG LIM IN TWIN PATH ___." Similar limits are attempted for paths with nonzero EI or EJ, since their flowrates will also depend on the enthalpy (density or temperature) changes within the tank. If this is the case, the message will read "FR CHG LIM, EI/EJ, PATH ___." Once again, time steps are never reduced by more than 50% due to this cause.

Other Limits—Often, the time step status line reports "TIME STEP GROWTH LIMIT." This message is due to the fact that the time step is constrained to grow by not more than 100% (a factor of two) each step. This limit means that some limit has previously held back the time step, but that it is now growing towards a new level. This previous limit is printed in the output routine header message. Also, for systems with no tanks or tubes, or for which the anticipated changes are negligible, the time step can be set very large and the message "NO LIMIT—TIME INDEPENDENT" printed. In practice, however, the user must intervene and place a reasonable upper limit on time step using DTMAXF.

Changes Made in User Logic—Only the effects of the latest pass through user logic blocks can be included in the time step controller; no ability exists to foresee future changes. Therefore, it is the user's responsibility to limit the time step (using DTMAXF or DTIMEH) as needed to capture any important variations invoked within the logic blocks. This caution applies to time– and temperature–varying parameters as well as valve closures, etc.

4.7.18.3 Control Constant Summary

Certain SINDA/FLUINT global constants (TIMEO, TIMEND, NLOOPS, ABSZRO) apply to fluid submodels also. Other global control constants apply only to fluid submodels: PATMOS, ACCELX, ACCELY, and ACCELZ. PATMOS (which is a double precision FORTRAN variable in logic blocks) is the pressure analog of ABSZRO. ACCELX/Y/Z is the body force acceleration vector, as described earlier.

Another class of control constants are output by the program for use by the user in the logic blocks. For fluid submodels, there is only one such constant for each submodel: DTIMUF, analogous to DTIMEU. Note that this variable is only current in FLOGIC 2 blocks; because the current time step cannot be computed until all user manipulations have been completed, the value of DTIMUF in FLOGIC 0 and 1 represents the previous time step; it is initialized to zero for the very first call to FLOGIC 0 or 1.

User-input, submodel specific control constants are also available. These variables are summarized in the bottom of Table 3–4. Note that each of these constants may vary by submodel.

DTMAXF—Maximum transient time step allowable. No fluid submodel will be integrated with a time step larger than DTMAXF. The default is 1.0E30 (essentially infinite). It is recommended that the default value *not* be used. A large characteristic limit should be used instead. Note that this is the fluid analog of DTIMEH, and that there is no analog for DTIMEI, which will be used as a maximum time step if fluid submodels are active.

DTMINF—Minimum *average* transient time step allowable. Transient integration will terminate after output calls if the time step is *persistently* below this value. The default is 0.0. Because of the relative volatility of fluid time step sizes, DTMINF is compared with a running average of the time step instead of the current time step. Note that this is the fluid analog of DTIMEL, although unlike that constant, DTMINF tolerates occasional wandering below the limit. No checks are performed in the first 20 time steps, and the program will only halt if the average time step is below the limit and does not appear to be increasing.

DTSIZF—Time step size factor. This parameter can be made to vary from 0.0 to 1.0 (*exclusive*) to control the size of the transient time step. The default is 0.1, which means that no important (time-dependent) network variable should change by more than about 10% during the next time step, or that no variable will change more than about 100.0*0.1/(1.0-0.1) = 11% past a transition. Increasing DTSIZF (say to 0.2 or 20%) will result in faster and probably less accurate solutions, while decreasing DTSIZF will result in slower and more accurate solutions. The user should be warned that instabilities may result if DTSIZF is too large (above about 0.5)—*increasing DTSIZF can sometimes even cause smaller time steps to be taken* for this reason. If there are no time-dependent elements, a time step of DTMAXF will be used for that submodel. The smallest time step of all active thermal or fluid submodels will be used.

DTTUBF—Time step size factor for tube flowrates. This parameter can be made to vary from 0.0 to 1.0 (*exclusive*) to control the size of the transient time step. The default is 0.1, which means that no tube flowrate (or slip flow velocity if twinned) should change by more than about 10% during the next time step. If tubes are being used for their stabilizing effects only, consider setting DTTUBF to 0.5 to allow greater time steps. DTTUBF is ignored if there are no tubes.

ITHLDF—Number of steady-state iterations a fluid submodel can stay dormant. This will be the fluid analog of ITHOLD. If a fluid submodel converges before other submodels, it is suspended from the solution sequence and held in a boundary state until other submodels catch up or ITHLDF iterations, whichever comes first. This suspension process is analogous to the actions of the DRPMOD/ADDMOD sequence—the submodel is present as a boundary condition for other submodels, but is not updated. ITHLDF=0 means no such suspension will occur. The default is 10. Like ITHOLD, *this is purely a CPU time savings measure, and its use usually results in an increase in the number of total iterations (LOOPCT) required even though total CPU time decreases since the average cost per iteration decreases.* The suspension and reactivization of fluid and thermal submodels are handled together, such that it is possible (and in fact usual) to have only one fluid and/or thermal submodel active during any one FASTIC or STDSTL iteration.

ITROTF—Steady-state iteration output interval. Analogous to ITEROT, this is the number of steady-state iterations between calls to output routines. A value of 0 will cause output routines to be called only after STDSTL is finished. A useful debugging trick is to set ITROTF to 1 in the last few iterations:

```
IF(LOOPCT .GE. NLOOPS - 10) ITROTF = 1
```

This trick may help explain the behavior of models that refuse to converge.

OPITRF—Transient output flag analogous to OPEITR. Setting OPITRF=1 will produce output at each time step. This should be used for debugging purposes only because of the large resultant files. For example, OPITRF might be turned on if the time step falls below an acceptable level. (DTMINF fulfills a similar purpose.)

OUTPTF—Output time interval for transient runs, analogous to OUTPUT for thermal submodels. Time steps will be adjusted if necessary to produce results at the times indicated by OUTPTF or OUTPUT.

REBALF—Relative error acceptable in steady-state fluid solutions for submodel or lump energy balance. The default is 0.01 or 1%. This is a convergence criterion for fluid submodels in STDSTL and FASTIC. Steady-state relaxation steps will continue (subject to NLOOPS and other control constants) as long as the relative error in the submodel energy balance is greater than |REBALF|. This relative error is defined as the difference between energy inputs and outputs divided by the energy inputs (absolute value). Energy flow is calculated from lump QDOT values (whether user supplied or FLUINT supplied) and flowrate-enthalpy products in and out of plena. At steady-state, the net energy storage in the submodel should approach zero.

Nonpositive REBALF has special meanings. First, a zero value signifies that the energy balance criterion should be skipped. Second, negative values signal that the energy balance criterion should be performed on a lump-by-lump basis instead of a submodel-level basis.

RERRF—Relative error acceptable in steady-state fluid solutions for key variables. The default is 0.01 or 1%. This is a convergence criterion for fluid submodels in STDSTL and FASTIC. Steady-state relaxation steps will continue (subject to NLOOPS and other control constants) as long as the relative change in value of key network variables is greater than RERRF. Key network variables include path flowrates and pressure differentials, and lump states. Relative error is defined as the change in value between relaxation steps divided by the present value (absolute value).

RSMAXF—Maximum artificial time step allowable with STDSTL. This is the steady state analog of DTMAXF. No fluid submodel will be integrated with an artificial time step larger than RSMAXF. The default is one hour. *It is highly recommended that the default value not be used*, especially for networks with many time-independent elements. A smaller characteristic value should be used instead.

RSSIZF—Steady-state relaxation step size factor. Analogous to DTSIZF, RSSIZF governs the allowable change in important network parameters (that would be time-dependent during a transient) during any one steady-state relaxation step in STDSTL. The default is 0.5 (50% change), although it can vary from 0.0 to 1.0 (*exclusive*). The user should use a small value if many alterations are made in the user logic blocks or nonconvergence is observed, and may use a large value for faster convergence if few alterations are made. RSSIZF is ignored if there are no time-dependent elements. If no time-dependent elements are present, an artificial time step of RSMAXF will be used. Because these time steps are artificial, different submodels are integrated with different time steps, which is not true for transient runs.

RSTUBF—Steady-state relaxation step size factor for tube flowrates. Analogous to DTTUBF, RSTUBF governs the allowable change in tube flowrates (that would be time-dependent during a transient) during any one steady-state relaxation step in STDSTL. The default is 0.5 (50% change), although it can vary from 0.0 to 1.0 (*exclusive*). RSTUBF is ignored if there are no tubes.

4.7.19 FLUINT Trouble Shooting

It is much easier to make a nonsensical fluid model than a nonsensical thermal model. Unfortunately, FLUINT cannot always determine whether or not a model makes physical sense until it attempts to solve it, and the failure modes of nonsensical models do not always leave clues as to what modeling error was made. This section describes common mistakes and modeling problems encountered with FLUINT. See also Section 4.7.16, Modeling Tips for Efficient Use.

Property Routine Errors—Excursions beyond the limits of the property library may cause either temporary or fatal problems. (Note that error messages indicating such problems will occasionally use English units even if SI was specified since many standard library fluid internal routines work strictly in English units.) Because such excursions are common during steady-state runs, FLUINT will first attempt to reset the properties to their limits and continue. However, too many or too severe errors will force an abort. Before suggesting ways of avoiding these limits, the reader should be familiar with the property range limits (see Appendix C for library fluids).

Some of the common sources for these errors (assuming that the system is not expected to operate outside of the range limits and is not overconstrained) include:

1) Large flow resistances (long lengths, small diameters or flow areas, large K factors) combined with unyielding pump models including VFRSET and MFRSET connectors, or overconstained flowrates. In such cases, the pressures may remain above or below acceptable limits in order to attempt to force the flow through the sections with large resistances. Excessively large lengths (TLEN values) alone can cause these problems for compressible fluids with UPF less than unity: the axial resolution is too coarse to resolve the internal pressure gradient.

2) Constant heat source terms combined with inadequate flowrates. If the QDOT value on a lump is nonzero and the flowrate through the lump becomes too small, the fluid in the lump will either "freeze" or become excessively superheated. This error is common if flowrates drop suddenly in tied junctions, as noted below, or if the flowrates oscillate by too large a magnitude. Flowrates can be steadied by eliminating competition among parallel paths, by adding tubes to constrain the changes, etc.

   If the flowrates are much lower than expected, check for (1) excessive resistances or large resistances compared to parallel paths, (2) pump degradation because of two-phase flow, and (3) depriming of a capillary device. Note that VFRSET, PUMP, and VPUMP models will quickly degrade if vapor is present at the inlet. While this is a realistic concern in most transients, it may be simply a pest in some steady-state runs. In the latter case, use a plenum or call HLDLMP to maintain subcooling in the inlet lump or use an MFRSET model. Also, a CAPIL connector with a small or zero RC can block vapor from the pump inlet as does a gas trap or filter in a real system.

3) All or a portion of the network has a defined, constant volume (even though it may be two-phase) and heat is added or subtracted until a pressure over or under limit is reached. Again, this may correspond to an actual problem in the real system.

4) An invalid state has been specified in a call to CHGLMP or in FLOW DATA. Check current values of ABSZRO and PATMOS. For FLOW DATA errors check PLADD, PLFACT, TLADD, and TLFACT, and check to see what the default PL, TL, and XL

values had been defined in previous LU DEF subblocks. Another common source of related errors is calling property routines (e.g., VSV and VTS) with values of temperature and pressure that are not in absolute units, or with fluid identifiers (the last argument) improperly defined or not allowed to be translated.

The user should note that "temperature too high" and "temperature too low" messages can be monotonously common for tied junctions when the flowrate can vary, especially in two-phase flow where heat rates are high and flow resistances can change quickly. These can be ignored unless they are persistent.

Oscillations and/or Hysteresis—There are many possible ways of arriving at different solutions when starting from different initial conditions, or getting long-term oscillations in the solution. The causes of these phenomena are sometimes physically based (as with capillary devices) and are sometimes artificially induced by mathematically sharp limits or one-to-many relationships that frequently occur in pressure drop and heat transfer correlations. As noted in Section 4.7.13, some causes are introduced purposely to enhance stability.

By the nature of its fully implicit solution and by using small artificial hysteresis in certain device models, FLUINT avoids most short term oscillations. Most of the large term oscillations and hysteresis are either physically based or are usually of no consequence to the analysis requirements. However, a few of the causes of these effects are outlined here for the user's understanding. More importantly, the user should be aware of similar effects caused by his own logic.

1) Capillary devices (CAPIL connectors and CAPPMP models) may exhibit fundamentally different behavior depending on the presence or absence of a capillary interface. Once deprimed, certain conditions must be met in order to reprime as is the case with real systems. This may be the cause of arriving at different answers from different directions. Note also that FLUINT uses small artificial hysteresis in the capillary pressure limit and the tank qualities to maintain stability over the short term. This hysteresis is on the order of 4% (real devices can exhibit hysteresis on the order of 50%), and hence should not be noticeable under normal circumstances. The routine SPRIME called from FLOGIC 0 can be used to force the device to stay primed, and the RC, XVH, and XVL descriptors of CAPILs and CAPPMP can be adjusted to make deprime less likely. One common cause of transient oscillation is the deprime of a capillary device due to excess pressure gradient where the liquid side is a subcooled tank. Refer to Section 4.7.7 and 4.7.8 for more details.

2) Heat transfer and pressure drop correlations can cause both hysteresis and oscillations. For example, accurate modeling of the transition region for single-phase flow means that one friction factor can be achieved at three different Reynolds numbers: (1) a laminar and (2) a turbulent number where small increases in Reynolds number corresponds to decreases in the friction factor, and (3) a transition number where the friction factor increases with increasing Reynolds number. While this presents no problem for quasi-steady solutions, it may cause some oscillations for fast changing solutions. As another example, heat transfer coefficients are orders-of-magnitude greater in two-phase regimes than single-phase regimes. Therefore, when heated past the saturation limit, a lump will tend to stay in the two-phase regime because of the greater amount of heat acquired in this mode. The converse is true for condensation—some oscillations may occur as a lump transitions back and forth

between a fast-cooling two-phase state and a slow-cooling single-phase state. Again, such effects may be visible during fast changes and steady-state solutions, but usually damp out for gradual changes.

3) In diabatic two-phase ducts, the spatial accelerations (forces due to flow area and density gradients) can cause long term oscillations, especially when such ducts are in parallel. Density (void fraction) oscillations are also common when using twinned paths. These accelerations are responsible to most oscillatory behavior in real evaporators and condensers. However, the user should not take such oscillatory predictions too literally because of the many approximations inherent in such analyses.

3) As described in Section 4.7.13, flow regime determination logic that is invoked by specifying IPDC=6 favors the current regime when there is doubt. This can result in substantial hysteresis.

4) Transitions from two–fluid to one–fluid modeling (slip flow to homogeneous with twinned paths, and nonequilibrium to perfectly mixed with the NONEQ0/1 routines) purposely contain hysteresis for stability.

Many problems with oscillations in two-phase modeling can be avoided (1) by using UPF=0.5 instead of 0.0 or 1.0, (2) by using C options instead of U or D in LINE and HX macros, (3) by using duplication options to avoid resolving interactions amongst parallel passages, and (4) by avoiding twinned paths if slip flow modeling is not necessary. The first two modeling decisions cause pressure drop and/or heat transfer calculations to be based on more than one lump or path, thereby lessening the impact of a change in one element and avoiding analytical trouble caused by severe gradients. The remaining decisions avoid unnecessary detail.

Many oscillations can be caused by rough user logic. Refer to the previous section on Modeling Tips for more details.

Matrix Solution Errors (YSMP)—Errors encountered during matrix inversion have error messages containing "YSMP" (Yale Sparse Matrix Package). This usually signals that at least one portion of the network is over or under constrained (see "Smoothing sensitive systems" under Modeling Tips). The cause of the error is printed in terms of the equation (mass or energy) and the lump in which the problem was detected.

Such errors can occur when all or an isolated portion of the network contains only hard-filled rigid tanks (COMP=XL=0.0) and/or junctions (including junction loops), or when the flowrates are overconstrained by too many "stiff" (GK=0.0) connectors. A plenum or "soft" tanks (COMP or XL > 0.0) must be present to provide a reference pressure or to absorb volume changes. At least one plenum is required for single-phase liquid models. Note that while an error message will be written if the entire network encounters such a condition, no such predictions are possible when a portion of the network is isolated from an otherwise satisfactory network by closed valves or other stiff connectors, as described below.

An isolated portion of the network is not just a region completely surrounded by closed valves. It is more loosely defined as any section of the network whose boundaries have prescribed flowrates (which includes the case of a prescribed zero flowrate corresponding to a closed valve). Paths that can prescribe flowrates include VFRSET and MFRSET connectors,

CTLVLV and CHKVLV connectors when closed, and any primed capillary connector (CAPIL connectors, NULL connectors in CAPPMP models). Such connectors are said to be *stiff* because their flowrate is insensitive to pressure gradient changes: GK = 0.0.

Examples of such overconstraining include more than one of the above connectors on a single line with no side branches, plena, or soft tanks in between. Note that a three-way control valve model should always leave one branch unconstrained to avoid such problems.

Memory Limitations—The YSMP package may also fail if it runs out of memory. Memory is allocated statistically and conservatively by the preprocessor, but the exact amount required cannot be known beforehand and can vary significantly with seemingly minor model changes. If the YSMP package aborts for this reason, a call to the reporting routine YSMPWK will be made and the user will be notified of the procedure required for avoiding this failure in future runs. This procedure involves overriding the preprocessor estimate with the NWORK keyword in the HEADER FLOW DATA subblock.

If the user believes memory allocations are excessive and is willing to assume the increased risk of an aborted run, the user can assume responsibility for YSMP memory allocation. By placing a call to the YSMPWK reporting routine (which has no arguments) *at the end of OPERATIONS DATA*, the user will be advised of actual memory usage. However, the user is forewarned that minor model changes can have a large impact on memory usage, and that FASTIC typically requires a small fraction (about a third) of the memory required by other solution routines.

4.8 Controlling Submodel States

One of the most powerful enhancements of SINDA/FLUINT over prior versions of SINDA is the ability to use submodels. **Users of old SINDA should pay particular attention to this section!** Submodels are not simply organizational tools, nor are they merely methods for avoiding node and conductor number collisions. The user can perform many manipulations at the submodel level, permitting a wide range of analyses not possible with old SINDA. Many clever tricks are available to the user who masters these manipulations.

While this material is also presented elsewhere in several sections, it is important enough to be collected and summarized in this section. Refer also to the sample problems in Appendix F, especially the first sample problem dealing strictly with thermal submodels.

BUILT vs. NOT BUILT—The BUILD and BUILDF commands define the list of currently active thermal and fluid submodels, respectively. This list is called the *configuration.* These commands may be repeated as necessary within OPERATIONS DATA.

Any submodel not built into the current configuration is treated as an adiabatic boundary condition (e.g., as if it does not exist) from the viewpoint of any other submodel referencing nodes or lumps in that submodel. Any submodel that is *never* built is treated as if it were never input—the input blocks for that submodel are not even preprocessed and its arrays, constants, and network parameter data (T, PL, UA, etc.) are not available. If the submodel is eventually built but is not present in the current configuration, its arrays and data are available in logic blocks but are not updated or accessed by any solution routines. Thus, the first time a submodel is built, it will contain all values input in the data blocks. The next time a submodel is built it will contain the values it had from the last time it was built. The only way to remove a submodel from the current configuration is to build a new configuration that omits it.

DORMANT vs. NOT DORMANT—Thermal submodels may be dynamically placed in a dormant state ("dropped") or reawakened ("added") using the DRPMOD and ADDMOD routines. DRPMOD (Section 6.6.10) is used to drop a submodel from the calculation sequence; subroutine ADDMOD (Section 6.6.11) reverses that action. Dormant models are not the same as inactive models (i.e., those not currently built). Whereas nodes in inactive models do not exist from the viewpoint of active models, nodes in dormant models are treated by other submodels as if they were boundary nodes. Hence, a dormant model may be said to be in a boundary state. Like BUILD and BUILDF commands, DRPMOD and ADDMOD may only be used in OPERATIONS DATA. (Caution: the actions of DRPMOD are not preserved from previous runs—the actions of RESTAR or RESTNC do not affect dormant status.)

Note that this dormant state is the same state used in steady-state calculations to minimize CPU time by dropping converged submodels from the solution sequence (see ITHOLD and ITHLDF control constants).

There is no fluid submodel equivalent to DRPMOD and ADDMOD. However, the HLDLMP and RELLMP routines perform an analogous function on a per-lump basis and may be called each iteration (rather than only within OPERATIONS DATA). There is no thermal submodel equivalent to these routines. Of course, an entire fluid submodel could be placed in a boundary state with repeated calls to HLDLMP, and a single node could be placed in a boundary state by isolating it in one submodel and then calling DRPMOD.

4.9 Introduction to the External Programs

SINDA/FLUINT consists of a preprocessor executable and a preprocessor load module. Additional external programs are available for either specialized pre- preprocessing or post-processing. The current SINDA/FLUINT external program library contains three programs: (1) EXPLOT, a graphics program designed primarily to generate x-y plots of all time-dependent program variables using the DISSPLA package on VAX hosts; (2) DATA_ONLY, a subset of EXPLOT that can be used to produce ASCII tabulations of SAVE files for possible use in other graphics packages; and (3) RAPPR, a program that helps the user create fast, custom, and reusable FPROP DATA blocks based on the standard library fluids.

Although still available, EXPLOT and DATA_ONLY have been rendered obsolete by the creation of SINAPS (SINDA Application Programming System), an interactive graphical pre– and postprocessor for SINDA/FLUINT. A brief introduction to SINAPS is given in the preface to this manual.

Please refer to Section 7 for detailed guidance in the use of all of the external programs. This section is strictly an overview.

4.9.1 EXPLOT: External Plotter

EXPLOT may be used to produce plots of data stored in SAVE files. (Restart files may be used equivalently.) EXPLOT provides extensive data presentation capability, featuring such options as comparative temperature histories from two or more parametric runs presented on a single plot grid. Another feature is the choice of one to four plot grids per 8 1/2 X 11 page. If an interactive graphics terminal is available, EXPLOT output may be viewed on a CRT prior to creating hardcopy plots. EXPLOT is fully interactive from the input standpoint. Menus are used to guide and prompt the user during input operations. Pre-existing input files may be edited and a "brief" input menu is featured that allows the user to quickly enter the most frequently used input parameters without having to respond to queries about the complete list of EXPLOT features.

4.9.2 DATA_ONLY: ASCII Tabulations from SAVE files

For those user who lack access to DISSPLA, DATA_ONLY may be used as an alternative to EXPLOT. Because DATA_ONLY produces ASCII tabulations of requested SAVE file data, it may also be used as a replacement for standard output routines. When used as such, DATA_ONLY is unique in that the user may decide *after* an analysis which data are important. However, its primary purpose is to allow users to interrogate SAVE files, creating data to be ported to other machines and/or other graphics packages.

4.9.3 RAPPR: Rapid Alternative Properties for Library Fluids

Full descriptions of the 20 standard library fluids can be slow to execute compared to simplified descriptions. Such simplified descriptions are valid over a necessarily smaller range than the full library descriptions, but few analyses require those full ranges. RAPPR provides the

user with the ability to create fast alternate descriptions of the library fluids over a smaller, customized range. These descriptions are in the form of reusable FPROP blocks, and may be created for all gas (8000 series), all liquid (9000 series), or local two-phase (6000 series). The range limits are indirectly input by the user in the form of allowable errors in properties (compared to the library description).

For water, ammonia, and common cryogenic fluids, alternative 6000 series tabular descriptions are available which execute as fast as RAPPR–generated 6000 series descriptions, but which have wider ranges and better accuracy over those ranges.

5. REFERENCE SUMMARY

This section is intended to serve as a fast look-up list of formats for experienced users. Figure 5–1 displays all possible input blocks that comprise an input file for SINDA/FLUINT. Note that the blocks may appear in any order with the exceptions of the OPTIONS DATA block, which must appear first, and the global CONTROL DATA and USER DATA blocks (if any) which must appear before submodel-specific CONTROL DATA and USER DATA blocks (if any). The blocks are shown in the order in which they are preprocessed.

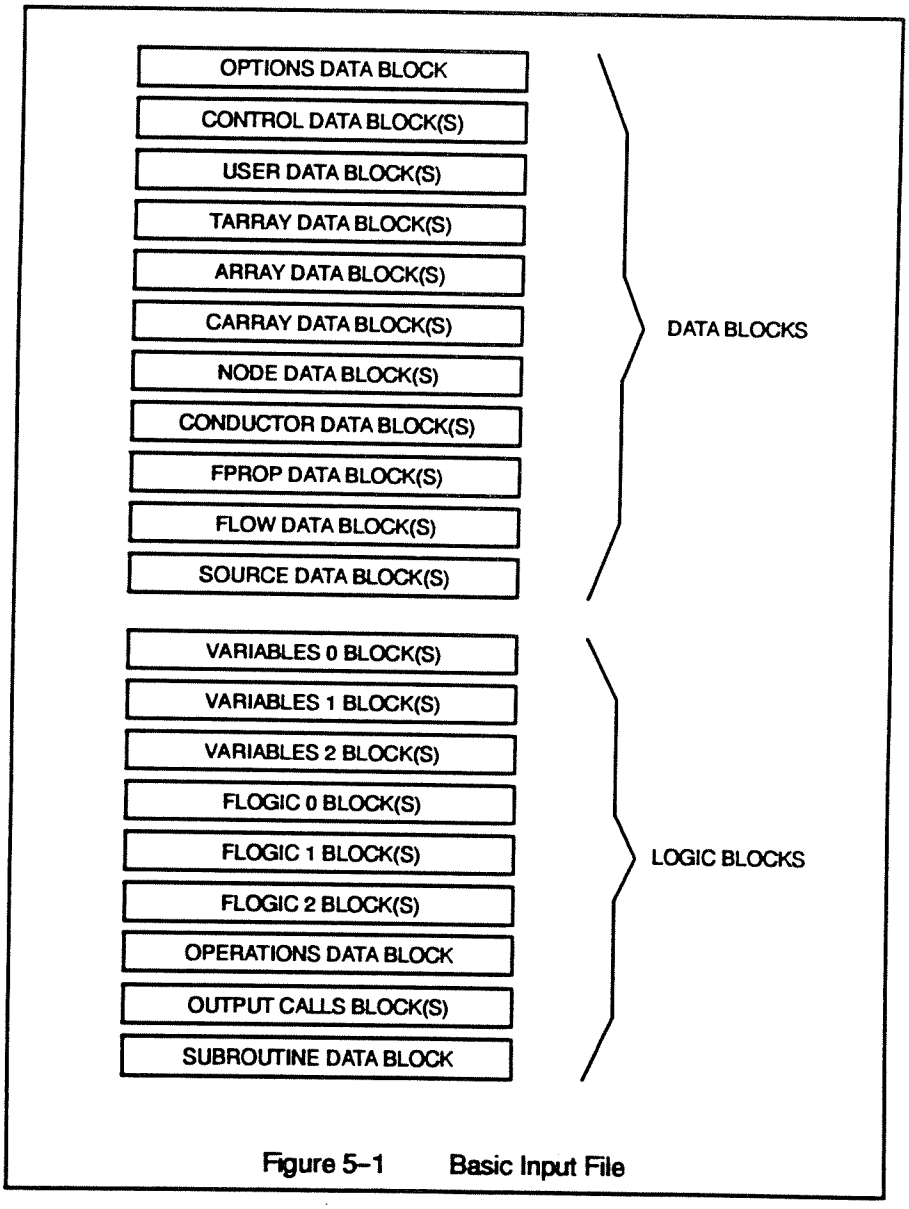


Figure 5–1 Basic Input File

5.1 Basic Conventions - Data Blocks

Data Types:

INTEGER ..... up to 10 numeric characters
FLOATING POINT ..... up to 20 numeric characters
CHARACTER ..... up to 8 characters (this doesn't include CARRAY data, which may be 128 characters long)

Comments:

```
      code    data-values $ comments
REM   comments
C     comments
```

5.2 Options Data

Basic Format:

```
HEADER OPTIONS DATA
     options data cards
```

Options Data Example:

```
HEADER OPTION DATA
TITLE    SAMPLE OPTION DATA
    PPSAVE     =    SAMPLE.PP
    RSI        =    SAMPLE.RSI
    RSO        =    SAMPLE.RSO
    OUTPUT     =    SAMPLE.OUT
    SAVE       =    SAMPLE.SAV
    QMAP       =    SAMPLE.QMP
    PLOT       =    SAMPLE.PLT
    USER1      =    SAMPLE.USR
    MODEL      =    SAMPLE
    PPOUT      =    ALL1
    NOLIST
    MLINE      =    66
    NODEBUG
    DIRECTORIES
```

Options Data Summary

Variable Name	Description and Options	Default
TITLE	Problem title– up to 72 alphanumeric characters	None – mandatory input
PPSAVE	Input save file – any permanent file name (pfn) up to 8 characters	None – optional input
RSI	Restart input file – any pfn	None – optional input
RSO	Restart output file – any pfn	None – optional input
OUTPUT	Processor print file – any pfn	Controlled by JCL (same as preprocessor print file) System print file name
SAVE	File for TSAVE & SAVE data – any pfn	None – optional input
QMAP	File for QMAP and FLOMAP data – any pfn	None – optional input
USER1 USER2	User auxiliary files – any pfn	None – optional input
MODEL	Model name, up to 8 characters	'ROOT'
PPOUT	Input data print/save control flag. ALL1, ALL2, ACTIV1, ACTIV2	ALL2 – Save all input, expanded
NOLIST	No printout flag	None – optional input
MLINE	Number of printed lines per page (integer)	54 – Univac, 58 – VAX
NODEBUG	User logic debug flag	Off – optional input
DIRECTORIES	Print flag for node, conductor, lump, path, tie, array, and array, and constant directories	Off – optional input

5.3 Node Data

Block Format:

```
HEADER NODE DATA, smn [,tcon]
     node data cards
```

where:

smn ..... thermal submodel name
tcon ..... temperature conversion flag

Node Types:

Diffusion .... have a thermal capacitance, can store energy
Arithmetic ... have a zero capacitance
Boundary .... have a constant temperature
Heater ..... have a constant temperature

Restrictions:

a. All transient *thermal* problems must include at least one diffusion node.
b. Node ID numbers should not exceed 6 digits.
c. Up to 20000 nodes may be defined.

Reference Forms:

```
T(n),Tn  =  temperature
Q(n),Qn  =  source
C(n),Qn  =  capacitance
```

where:

n ..... actual node number, positive even when referencing boundary nodes.

Diffusion nodes have a T, Q and C.
Arithmetic and Heater nodes have a T and Q.
Boundary nodes have only a T.

Units:

Temperature . degrees
Source ..... energy/time
Capacitance . energy/degree

Node Data Input Options:

```
N#,Ti,C                                  $ Single node*

GEN N#,#N,IN,Ti,C                        $ Group of nodes*

SIV N#,Ti,AP,F                           $ C vs. T (interp)
SPV N#,Ti,AC,F                           $ C vs. T (poly)

SIM N#,#N,IN,Ti,AP,F                     $ GEN and SIV
SPM N#,#N,IN,Ti,AC,F                     $ GEN and SPV

DIV N#,Ti,AP1,F1,AP2,F2                  $ 2 mat'l (interp)
DIV N#,Ti,SUB,F1,AP2,F2                  $ 2 mat'l,C1=const
DIV N#,Ti,AP1,F1,SUB,F2                  $ 2 mat'l,C2=const

DPV N#,Ti,AC1,F1,AC2,F2                  $ 2 mat'l (poly)
DPV N#,Ti,SUB,F1,AC2,F2                  $ 2 mat'l,C1=const
DPV N#,Ti,AC1,F1,SUB,F2                  $ 2 mat'l,C2=const

DIM N#,#N,IN,Ti,AP1,F1,AP2,F2            $ GEN and DIM
DIM N#,#N,IN,Ti,SUB,F1,AP2,F2
DIM N#,#N,IN,Ti,AP1,F1,SUB,F2

DPM N#,#N,IN,Ti,AC1,F1,AC2,F2            $ GEN and DPV
DPM N#,#N,IN,Ti,SUB,F1,AC2,F2
DPM N#,#N,IN,Ti,AC1,F1,SUB,F2

BIV N#,Ti,AB,F                           $ C vs. T & time
```

* Note:

Diffusion .... positive node number, positive capacitance
Arithmetic ... positive node number, negative capacitance
Boundary .... negative node number, non-negative capacitance
Heater ..... negative node number, negative capacitance

where:

N# ..... Actual node number (non-zero integer)

T_i ..... Initial temperature (floating point)

C ..... Capacitance (floating point)

#N ..... Number of nodes (non-zero integer)

IN ..... Increment to the node number (non-zero integer)

AP_n ..... References to arrays of temperature vs. capacitance *points* (Integer count form)

AC_n ..... References to arrays of polynomial *coefficients* for C = P(T) (Integer count form), $[P_0, P_1, P_2, \ldots P_N]$

F_n ..... Multiplying factors (floating point data values, or references to user constants)

AB ..... Reference to a bivariate array of temperature and time vs. capacitance (Integer count form)

SUB ..... Constant value substitute for an array reference (floating point data value)

Summary of NODE DATA Input Options

OPTION (CODE)	NODE TYPE D	A	B	H	DESCRIPTION
3 blanks	■	■	■	■	To input a single node where the capacitance is given as a single, constant value
CAL	■				To input a single node where the capacitance will be calculated by the preprocessor from four factors input by the user
GEN	■	■	■	■	To generate a group of nodes, each having the same initial temperature and the same capacitance
SIV SPV	■				To input a single node where the capacitance varies with temperature. For SIV, the capacitance is found by interpolation using an array of temperature vs. capacitance. For SPV, the capacitance is found by computing an Nth order polynomial function of temperature
SIM SPM	■				To generate a group of nodes, each having the same initial temperature and the same temperature–varying capacitance. For SIM, C is found by interpolation using an array of T vs.C. For SPM, C is found by computing a polynomial in T
DIV DPV	■				To input a single node consisting of two materials which have different temperature–varying capacitances. For DIV, C1 and C2 are taken from arrays of T vs. C. For DPV, C1 and C2 are computed from polynomials in T
DIM DPM	■				To generate a group of nodes, each of which consists of the same two materials having temperature–varying capacitances. For DIM, C1 and C2 are taken from arrays of T vs. C. For DPM, C1 and C2 are computed from polynomials in T.
BIV	■				To input a single node where the capacitance is a function of time and temperature. The capacitance is found by interpolation using an array of time and temperature vs. capacitance

5.4 Source Data

Block Format:

```
HEADER SOURCE DATA, smn
     source data cards
```

Reference Forms:

```
Q(n),Qn
```

where n is the actual node number.

Units:

Source ..... energy/time

Source Data Input Options:

```
N#,Q

GEN N#,#N,IN,Q

TVS N#,AS,AQ,F                          $ Q vs. T

SIT N#,At,F                             $ Q vs. time

TVD N#,AS,AQ,FA,AS,AQ,FA                $ Q1+Q2 vs. time

DTV N#,At,F1,AT,F2                      $ Q vs. T & time
DTV N#,SUB,F1,AT,F2
DTV N#,At,F1,SUB,F2

PER N#,AS,AQ,F,D,P
```

where:

N# ..... Actual node number (positive integer)

Q ..... Value of source (floating point data value or reference to a user constant

#N ..... Number of nodes (non-zero integer)

IN ..... Increment to the node number (non-zero integer)

AT ..... Reference to an array of temperature vs. heat rate points (integer count form)

At_n ..... References to arrays of mean time vs. heat rate points (integer count form). The first time point must be zero

F_n ..... Multiplying factors (floating point data value, or reference to a user constant)

AS ..... Reference to a singlet time array

AQ ..... reference to a singlet Q array

SUB ..... Value used as a substitute for an array reference (floating point data value)

D ..... Phase displacement as a decimal fraction of the total period. (Must be a floating point literal with a value between O. and 1.0.)

P ..... Period-specified in time units (floating point data value or reference to a user constant). P is usually the last time point in the At array, but not restricted to this value. If P is less than the last array value, any values between P and the last array value are not used. If P is greater than the last array time value, the heating value used for the time period between the last array time value and P is constant and equal to the heating rate input for the last array time value.

Summary of SOURCE DATA Input Options

OPTION (CODE)	DESCRIPTION
3 blanks	To impress a constant heat source on a single node
GEN	To impress the same heat source on several nodes
SIV	To impress a temperature–varying heat source on a node
SIT†	To impress a time–varying heat source on a node
DIT†	To impress the sum of two time–varying heat sources on a node
TVS	To impress a time–varying heat source on a node
TVD	To impress the sum of two time–varying heat sources on a node
DTV	To impress the sum of a time–varying source and a temperature–varying source on a node
CYC†	To impress a cyclic time–dependent source on a node
PER	To impress a cyclic time–dependent source on a node

† Beginning with SINDA '85, the SIT, DIT, and CYC options have been replaced with the TVS, TVD, and PER options respectively. The obsolete options are included only for processing old SINDA files.

5.5 Conductor Data

Block Format:

```
HEADER CONDUCTOR DATA, smn [,fac]
     conductor data cards
```

Conductor Types:

LINEAR ..... heat transfer a function of (T_i - T_j)
RADIATION . heat transfer a function of (T_i^4 - T_j^4)

Restrictions:

a. Conductor number may be as large as six digits.
b. Up to 100000 conductors may be defined.

References:

```
G(n),Gn
```

where n is an actual conductor number (always positive)

Units:

LINEAR ..... energy/time-degree
RADIATION . energy/time-degree^4

Conductance Calculations:

Conduction .. kA/L
Convection ... hA
Radiation .... $\sigma\mathcal{F}A$
Mass Flow ... mC_p

where:

k ..... conductivity (energy/time-length-degree)
A ..... area (length^2)
L ..... length
h ..... film coefficient (energy/time-length^2-degree)
m ..... mass flowrate (mass/time)
C_p ..... specific heat (energy/mass-degree)
σ ..... Stefan-Boltzmann constant (energy/time-length^2-degree^4)
$\mathcal{F}$ ..... Hottel "script F" or grey body radiation interchange factor

Conductor Data Input Options:

```
G#,NA,NB,G                                  $ One conductor
G#,NA1,NB1,NA2,NB2,... NAn,NBn,G            $ Multi-connected
GEN G#,#G,IG,NA,INA,NB,INB,G                $ Group of cond.

SIV G#,NA,NB,AP,F                           $ G vs T (interp)*
SPV G#,NA,NB,AC,F                           $ G vs T (poly)*
SIM G#,#G,IG,NA,INA,NB,INB,AP,F             $ GEN and SIV
SPM G#,#G,IG,NA,INA,NB,INB,AC,F             $ GEN and SPV
                                            $ * (Note 1)

DIV G#,NA,NB,APA,FA,APB,FB                  $ 2 mat'l (interp)
DIV G#,NA,NB,SUB,FA,APB,FB                  $ 2 mat'l GA=const
DIV G#,NA,NB,APA,FA,SUB,FB                  $ 2 mat'l GB=const

DPV G#,NA,NB,ACA,FA,ACB,FB                  $ 2 mat'l (poly)
DPV G#,NA,NB,SUB,FA,ACB,FB                  $ 2 mat'l GA=const
DPV G#,NA,NB,ACA,FA,SUB,FB                  $ 2 mat'l GB=const

DIM G#,#G,IG,NA,INA,NB,INB,APA,FA,APB,FB    $ GEN & DIV
DIM G#,#G,IG,NA,INA,NB,INB,SUB,FA,APB,FB
DIM G#,#G,IG,NA,INA,NB,INB,APA,FA,SUB,FB

DPM G#,#G,IG,NA,INA,NB,INB,ACA,FA,ACB,FB    $ GEN & DPV
DPM G#,#G,IG,NA,INA,NB,INV,SUB,FA,ACB,FB
DPM G#,#G,IG,NA,INA,NB,INB,ACA,FA,SUB,FB

BIV G#,NA,NB,AB,F                           $ G vs T & time

PIV G#,NA,NB,AT,FT,AAT,FAT                  $ G=f(T & g(time))
PIV G#,NA,NB,SUB,FT,AAT,FAT                 $ (Note 4)
PIV G#,NA,NB,AT,FT,SUB,FAT

PIM G#,#G,IG,NA,INA,NB,INB,AT,FT,AAT,FAT    $ GEN & PIV
PIM G#,#G,IG,NA,INA,NB,INB,SUB,FT,AAT,FAT   $ (Note 4)
PIM G#,#G,IG,NA,INA,NB,INB,AT,FT,SUB,FAT

TUS G#,NA,NB,AS,AG,F

PER G#,NA,NB,AS,AG,D,P
```

where:

G# ..... Actual conductor number (non-zero integer) [Note 2]

NA,NB .... Actual node numbers of the nodes to which the conductor is connected (positive integer—see Note 3)

G ..... Conductor value (floating point data value)

W,X,Y,Z .. Values used to compute conductance: G = W*X*Y/Z (floating point)

#G ..... Total number of conductors to be generated (non-zero integer)

IG ..... Increment to the conductor number (integer)

INA,INB .. Increments to the node number, NA and NB, respectively (integer)

AP_A ..... Reference to arrays of temperature vs conductance POINTS (integer count form)

AC_A ..... References to arrays of polynomial *coefficients* for G = P(T).

F_A ..... Multiplying factors (floating point data values—see Note 1, or references to user constants)

SUB ..... Value used as a substitute for an array reference (floating point data value).

AT ..... Reference to doublet arrays of an arbitrary variable (A) versus time (τ). A = f(τ)

AAT ..... Reference to bivariate arrays of conductance (G) versus a temperature variable (T) and an arbitrary variable (A). G = f(T,A). The temperature used in calculation is the average of NA and NB. [Note 5]

AS ..... Reference to a singlet time array

AG ..... Reference to a singlet G array

FT,FAT ... Multiplication factors (floating point data values, or reference to user constants)

Notes:

1. Normally, the average temperature of nodes NA and NB is used to evaluate the conductance. To indicate that only the temperature of node NA is to be used, F must be entered as a floating point data value preceded by a minus sign. The minus sign is considered only as a flag and is ignored in the arithmetic sense. This feature applies only to the options noted.

2. Conductor type is specified by the sign of the G# as follows:

   LINEAR ..... positive conductor number
   RADIATION ..... negative conductor number

3. The negative signs on node numbers which define boundary nodes are considered flags and are ignored in the arithmetic sense. Preceding NA or NB with a minus sign will cause the conductor to be treated as a one-way conductor. The "upstream" node should be identified with the minus sign, and will be unaware of the presence of the "downstream" node, but not vice versa.

4. In both the PIM and PIV options,when:

   SUB,FT,AAT,FAT, is used, the arbitrary variable (A) is evaluated as A = SUB*FT.

AT,FT,SUB,FAT is used, the arbitrary variable (A) is evaluated by interpolation. A = f(e); the conductance (G) is evaluated as G = A*FT*SUB*FAT.

5. Consistent with the bivariate array input format described in Section 3, the arbitrary variable (A) corresponds to the Y values and the temperature (T) corresponds to the X values. The units of the arbitrary variable entered into the bivariate array must be equal to the product of the arbitrary variable and the multiplication factor (FT) for the time dependent doublet array.

Summary of CONDUCTOR DATA Input Options

OPTION (CODE)	DESCRIPTION
3 blanks	To input single conductors
CAL†	To input single conductors where G is a strict function of 4 factors
GEN	To generate several conductors with the same conductance
SIV SPV	To input single temperature–varying conductors. SIV uses array interpolation, SPV uses a polynomial function of temperature
SIM SPM	Same as SIV and SPV for groups of conductors
DIV DPV	To input single conductors that model two materials with different temperature–varying conductances. DIV uses two arrays of G vs. T, and DPV uses polynomial functions of T.
DIM DPM	Same as DIV and DPV for groups of conductors
BIV	To input a single conductor that is both time– and temperature–varying. The conductance is found by interpolation using a bivariate array.
PIV PIM	To input conductors where the conductance is a function of temperature and a time–independent variable. PIV is for single conductors, PIM is for multiple conductors
TVS	To input a time–varying conductor
PER	To input a cyclic time–varying conductor

† Beginning with SINDA '85, floating point arithmetic (+,–,*,/) in data fields makes the CAL option obsolete. The CAL option is available but not documented.

5.6 Control Data

Block Format:

```
HEADER CONTROL DATA, GLOBAL
     control data cards
HEADER CONTROL DATA, smn
     control data cards
```

Restrictions:

a. GLOBAL data must be entered before any submodel data.
b. Only control constants may be entered in this block.

Reference Forms:

variable = value

Control Data Input Options:

variable = value

User-Specified Control Constants Required by Solution Routines

Name	Steady State STDSTL, FASTIC	Transient FWDBCK	Transient FORWRD
TIMEO	0.0	0.0	0.0
TIMEND	0.0	0.0	0.0
NLOOPS	R,F	NA	NA
DTIMES	0.0	NA	NA
ABSZRO	-459.67 or -273.15*	-459.67 or -273.15*	-459.67 or -273.15*
PATMOS	0.0	0.0	0.0
SIGMA	1.0	1.0	1.0
ACCELX	0.0	0.0	0.0
ACCELY	0.0	0.0	0.0
ACCELZ	0.0	0.0	0.0
ARLXCA	0.01	0.01	0.01
NLOOPT	NA	100	100
DRLXCA	0.01	0.01	NA
EBALSA	0.01	NA	NA
EBALNA	0.0	NA	NA
ATMPCA	NA	1.0E30	1.0E30
DTMPCA	NA	1.0E30	1.0E30
CSGFAC	NA	NA	1.0
DTIMEL	NA	0.0	0.0
DTIMEH	NA	1.0E30	1.0E30
DTIMEI	NA	0.0	NA
ITEROT	0	NA	NA
OUTPUT	NA	R	R
ITHOLD	10	NA	NA
EXTLIM	50.0	50.0	50.0
ITERXT	3	3	3
BACKUP	0	0	0
OPEITR	NA	0	0
DTSIZF	NA	0.1	0.1
DTTUBF	NA	0.1	0.1
DTMAXF	NA	1.0E30	1.0E30
DTMINF	NA	0.0	0.0
OUTPTF	NA	F	F
OPITRF	NA	0	0
RSSIZF	0.5	NA	NA
RSTUBF	0.5	NA	NA
RSMAXF	1 hour	NA	NA
RERRF	0.01	NA	NA
REBALF	0.01	NA	NA
ITHLDF	10	NA	NA
ITROTF	0	NA	NA

where:

R ..... required value (if thermal submodels present)
NA ..... does not apply
F ..... required only if fluid submodels present
* ..... default depends on value of UID.

5.7 User Data

Block Format:

```
HEADER USER DATA, GLOBAL
     user data cards
HEADER USER DATA, smn
     user data cards
```

Restrictions:

a. GLOBAL data must be entered before any submodel data.
b. Only alphanumeric variables are allowed in the global block.
c. Integer variables must be initialized with integer data.
d. Character data is not allowed.
e. Only one GLOBAL block is allowed.
f. Variables that start with I-N will be integers, all other are real.
g. The total number of numbered constants must not exceed 1000.

User Data Input Options Example:

```
HEADER USER DATA, GLOBAL
     ITEST  = 1
     AB     = 2.0
HEADER USER DATA, SMN
     ID#    = DV
     ID#,DV
     GEN ID#,#K,IK,DV
     GEN ID#,#K,IK,DVI,IDV
```

where:

ID# ..... Integer identification number
#K ..... the number of constants to be generated
IK ..... ID# increment
DV ..... data value
SMN ..... a submodel name

Reference Forms:

```
Kn or K(n)          $ for integer reference
XKn or XK(n)        $ for real reference
```

5.8 Array Data

Block Format:

```
HEADER CARRAY DATA, smn
     carry data cards
HEADER TARRAY DATA, smn
     tarry data cards
HEADER ARRAY DATA, smn
     array data cards
```

Restrictions:

a. A combined total of 10,000 arrays may be entered into ARRAY and TARRAY DATA.
b. 1000 arrays may be entered into CARRAY DATA.
c. All arrays in TARRAY DATA must be singlet arrays.
d. Character data can only be entered in CARRAY DATA.
e. Hollerith data is not allowed.
f. Entries in CARRAY DATA are limited to 128 characters.
g. Maximum of 10,00 entries in any one numeric array.

Reference Forms:

```
NAn or NA(n)          $ for integer reference
AN  or A(n)           $ for real reference
UCAn or UCA(n)        $ for character reference
```

Note TARRAY data can not be directly referenced.

Array Data Input Options:

```
HEADER CARRAY DATA,smn
     array-number = a string
HEADER ARRAY DATA,smn
     array-number = dv1,dv2,dv3,dv4
HEADER TARRAY DATA,smn
     array-number = dv1,dv2,dv3,dv4
```

5.9 Logic Blocks

5.9.1 Block Formats

```
HEADER OPERATIONS DATA
     operations data statements
HEADER VARIABLES 0, smn
     variables 0 statements
HEADER VARIABLES 1, smn
     variables 1 statements
HEADER VARIABLES 2, smn
     variables 2 statements
HEADER FLOGIC 0, smn
     flow logic 0 statements
HEADER FLOGIC 1, smn
     flow logic 1 statements
HEADER FLOGIC 2, smn
     flow logic 2 statements
HEADER OUTPUT CALLS, smn
     output operations
HEADER SUBROUTINE DATA
     user subroutines
```

5.9.2 F-Type FORTRAN Statements

Permissible Arguments:

a. any acceptable FORTRAN statement.
b. T(m), Q(m), C(m), K(m), G(m), A(m), XK(m), NA(m), UCA(m) where m = any valid FORTRAN subscript, i.e., a relative number.
c. FLUINT variable names unless noted.

Record Format:

```
1      7
Fsn    statement
```

where `sn` is a statement number.

If a statement must be continued on a second card, this card must have any character, except a blank or a zero, in column 6.

5.9.3 M-Type FORTRAN Statements

Permissible Arguments:

a. Tn, T(n), Qn, Q(n), Cn, C(n), Kn, K(n), Gn, G(n), An, A(n), XKn, XK(n), NAn, NA(n), UCAn, UCA(n) and FLUINT names where n = actual number
b. any control constant name
any of the above arguments can (sometimes must) be preceded by a submodel name: smn.T(100) or smn.ARLXCA etc.
c. any data value
d. any named user constant
e. any argument permissible in FORTRAN
f. GLOBAL.reserved(m)
g. smn.reserved(n)
h. GLOBAL.reserved(m+e)
i. smn.reserved(n+e)

where:

reserved ..... T, Q, C, K, G, A, XK, NA, UCA, FLUINT variable, or control constant
e ..... any M-type expression
m ..... a relative index
n ..... an actual index

Record Format:

```
1       7
sn      statement
```

where `sn` is a statement number.

5.9.4 FLUINT Variables

Lumps:

All ..... PL (double precision), TL, HL, DL, XL, CX, CY, CZ
Junc/tank ... QDOT
Tank ..... VOL, VDOT, COMP

Paths:

All ..... FR, DUPI, DUPJ, GK, HK, EI, EJ, DK
Tube ..... AM
Tube/STUBE . HC, FC, FPOW, AC, IPDC, UPF, TLEN, DH, AF, AFTH, FK, WRF, FG, FD
MFRSET .... SMFR
VFRSET..... SVFR
Loss/valve ... FK, TPF, AF, AFTH
LOSS2 ..... FK, TPF, AF, AFTH, FKB
CAPIL ..... RC, CFC, XVH, XVL (same for 1st CAPPMP connector)
VPUMP ..... SPD

Ties:

All ..... UA, QTIE, DUPL, DUPN

FLUINT Utility Routines:

CHGLMP.... Change lump PL, TL, XL
CHGVOL .... Change tank VOL
CHGSUC .... Change path suction status
HLDLMP .... Make a tank or junction act like a plenum
HTRLMP .... Holds the enthalpy of a junction or FASTIC tank
RELLMP .... Reverse action of HLDLMP and HTRLMP
GOHOMO ... Force one or all twinned paths to stay homogeneous
GOSLIP ..... Reverse action of GOHOMO: enable slip flow in twins
INTLMP .... Fetch internal lump sequence number
INTTIE ..... Fetch internal tie sequence number
INTPAT ..... Fetch internal path sequence number
PUTTIE ..... Move a tie to a new node and/or lump
SPRIME..... Keep a capillary device primed
XTRACT .... Calculate effective quality extracted by a path

5.9.5 Reserved Words

This subsection summarizes the reserved word list. See also Section 3.12. Reserved words are SINDA/FLUINT variables and arrays that are contained in common blocks. These common blocks are inserted automatically into the routines resulting from logic blocks such as OPERATIONS DATA, OUTPUT CALLS, VARIABLES 0/1/2, and FLOGIC 0/1/2. They are only inserted into routines in SUBROUTINE DATA if the CALL COMMON command is used. *It is the user's responsibility to avoid collisions with the reserved words.* In this context, a "collision" is the inadvertent use of a reserved word as a user variable or array name.

The reserved word list consists of all translatable network parameters and control constants, *but not all reserved words are translatable.* Table 5–1 lists all reserved words in alphabetical order. In runs with no fluid submodels, the reserved word list is reduced. However, it is considered good practice to avoid FLUINT reserved names in case fluid submodels are added later.

Table 5–1 Reserved Word List

A	DUPN	MXNODE	NNODE	NZZZZ2
ABSZRO	EBALNA	MXTARY	NOARRY	NZZZZ3
AC	EBALNC	MXTOKN	NOCARY	OPEITR
ACCELX	EBALSA	MXUARY	NOCNTR	OPITRF
ACCELY	EBALSC	MXUCON	NOOPT	OUTPTF
ACCELZ	EI	NA	NOSORC	OUTPUT
AF	EJ	NARLXC	NOTARY	PAGECT
AFTH	ESUMIS	NARLXN	NOUSE	PATMOS
AM	ESUMOS	NATMPC	NOUSER	PG
ARLXCA	EXTLIM	NATMPN	NOUT	PL
ARLXCC	FC	NCGMAC	NPARAM	PT
ATMPCA	FD	NCGMAN	NQMP	Q
ATMPCC	FDATA	NCOND	NRAD	QDOT
BACKUP	FG	NCSGMC	NRSI	QTIE
C	FI	NCSGMN	NRSO	REBALF
COMP	FK	NCTOT	NSCR1	RERRF
CSGFAC	FLID	NDINT	NSCR2	RSMAXF
CSGMAX	FPOW	NDNAM	NSCR3	RSSIZF
CSGMIN	FR	NDRLXC	NSCR4	RSTUBF
CX	G	NDRLXN	NSOL	SIGMA
CY	GK	NDTMPC	NSTRT	T
CZ	HC	NDTMPN	NTC	TIMDY
DATE	HK	NEBALC	NTJ	TIMEM
DH	HL	NEBALN	NTK	TIMEN
DK	IPDC	NETQF	NTL	TIMEND
DL	ITEROT	NFLO	NTOTK	TIMEO
DRLXCA	ITERXT	NFLOW	NTOTL	TL
DRLXCC	ITHLDF	NFLOW2	NTPL	TLEN
DTIMEH	ITHOLD	NFLOW3	NTSV	TOLD
DTIMEI	ITROTF	NFM	NTT	UA
DTIMEL	K	NFPROP	NTTB	UCA
DTIMES	LINECT	NGSTRT	NUMCNT	UID
DTIMEU	LOOPCT	NIN	NUMJNC	UPF
DTIMUF	MLINE	NLIN	NUMPLE	VDOT
DTMAXF	MMODS	NLOOPS	NUMTNK	VOL
DTMINF	MXCNTR	NLOOPT	NUMTUB	VPGMAX
DTMPCA	MXCOND	NMACT	NUSER1	VPS
DTMPCC	MXFILE	NMARI	NUSER2	WRF
DTSIZF	MXFLID	NMBD	NVCD	XK
DTTUBF	MXFRES	NMDIF	NVGD	XL
DUPI	MXMODF	NMHT	NXTRA1	
DUPJ	MXMODS	NMOD	NXTRA2	
DUPL	MXNCON	NNOD	NZZZZ1	

5.10 Flow Data

Block Format:

```
HEADER FLOW DATA,smn[,FID=I][,NWORK=I]
     flow data cards
```

FLOW DATA Restrictions:

a. Lump and path numbers should be 6 digits or less.
b. Up to 4000 lumps, 6000 ties, and 4000 paths may be defined.
c. Tanks and junctions must be connected to at least one but no more than 25 other lumps.
d. No model can be built entirely of junctions.
e. No loop of diabatic junctions can exist.
f. Net flow through junctions must be nonzero if the QDOT is nonzero.
g. Model must contain at least one plenum, vapor-filled or compliant tank.
h. Flowrate cannot be overconstrained along any line or through any junction.
i. PL, TL, VOL, FR, TLEN, DH have no defaults until set in a DEF subblock.
j. Excluding DUP factors and plena, a maximum of 25 paths are allowed per lump

5.10.1 LU (Lump) Subblocks

Lump Types:

Tank ..... Finite volume (resists mass and energy changes)
Junction ..... Zero volume (state can change instantaneously)
Plenum ..... Infinite volume (constant state)

Lump Subblock Formats:

```
LU TANK,id[,VOL=R][,TL=R][,XL=R][,PL=R][,PL!=R]
     [,VDOT=R][,QDOT=R][,COMP=R]
     [,CX=R][,CY=R][,CZ=R]

LU PLEN,id[,TL=R][,XL=R][,PL=R][,PL!=R]
     [,CX=R][,CY=R][,CZ=R]

LU JUNC,id[,TL=R][,XL=R][,PL=R][,PL!=R][,QDOT=R]
     [,CX=R][,CY=R][,CZ=R]

LU DEF[,VOL=R][,TL=R][,TLADD=R][,TLFACT=R]
     [,XL=R][,PL=R][,PL!=R][,PLADD=R][,PLFACT=R]
     [,VDOT=R][,QDOT=R][,COMP=R]
     [,CX=R][,CY=R][,CZ=R]
```

5.10.2 PA (Path) Subblocks

Path Types:

Tube ..... Finite inertia (resists flowrate changes)
Connector ... Zero inertia (flowrate can change instantaneously), subclassified by device specification.

Path Subblock Formats:

```
PA TUBE,id,l1,l2[,FR=R][,TWIN=I][,STAT=C]
      [,TLEN=R][,DH=R][,AF=R][,AFTH=R][,FK=R]
      [,WRF=R][,IPDC=I][,UPF=R]
      [,HC=R][,FC=R][,FPOW=R][,AC=R][,AM=R][,FG=R][,FD=R]
      [,DUPI=R][,DUPJ=R][,DUP=R]

PA CONN,id,l1,l2[,FR=R][,TWIN=I][,STAT=C]
      [,GK=R][,HK=R][,EI=R][,EJ=R][,DK=R]
      [,DUPI=R][,DUPJ=R][,DUP=R]
      [,DEV=C,...,...]

PA DEF[,FR=R][,FRFACT=R][,STAT=C]
      [,TLEN=R][,DH=R][,AF=R][,AFTH=R]
      [,WRF=R][,IPDC=I][,UPF=R]
      [,HC=R][,FC=R][,FPOW=R][,AC=R][,AM=R][,FG=R][,FD=R]
      [,GK=R][,HK=R][,EI=R][,EJ=R][,DK=R]
```

Device Formats within CONN Subblocks:

```
... DEV=STUBE [,TLEN=R][,DH=R][,AF=R][,AFTH=R][,FK=R]
              [,WRF=R][,IPDC=I][,UPF=R]
              [,HC=R][,FC=R][,FPOW=R][,AC=R][,FG=R][,FD=R]

... DEV=CAPIL,RC=R,CFC=R[,XVH=R][,XVL=R]

... DEV=LOSS,FK=R[,AF=R][,AFTH=R][,TPF=R]

... DEV=LOSS2,FK=R[,FKB=R][,AF=R][,AFTH=R][,TPF=R]

... DEV=CHKVLV,FK=R[,AF=R][,AFTH=R][,TPF=R]

... DEV=CTLVLV,FK=R[,AF=R][,AFTH=R][,TPF=R]

... DEV=MFRSET[,SMFR=R]

... DEV=VFRSET,SVFR=R

... DEV=PUMP,HG=A[,DEGVF=A]

... DEV=VPUMP,HG=A,RSPD=R[,SPD=R][,DEGVF=A]
```

5.10.3 T (Tie) Subblocks

Tie Types:

HTN ..... Links lump to node with all or part of one path representing size and shape for convection heat transfer; lump represents average fluid state

HTNC ..... Links lump to node with all or parts of two paths representing size and shape for convection heat transfer (centered lump); lump represents average fluid state

HTU ..... Links lump to node with user-specified UA conductance; lump represents average fluid state

HTNS ..... Links path to node representing average wall temperature over that segment, path represents size and shape for convection heat transfer; heat applied to downstream lump

HTUS ..... Links path to node representing average wall temperature over that segment, user–described heat transfer coefficient; heat applied to downstream lump

Note: Any named path can also represent the primary twin of a pair of twinned paths, in which case the involvement of the secondary path is implied. Secondary twins, however, cannot be named directly.

Tie Subblock Formats:

```
T HTN,tid,lid,nid,pid[,f][,DUPN=R][,DUPL=R]

T HTNC,tid,lid,nid,pid,f1,p2id[,f2][,DUPN=R][,DUPL=R]

T HTU,tid,lid,nid,UA=R[,DUPN=R][,DUPL=R]

T HTNS,tid,pid,nid,[,f][,DUPN=R][,DUPL=R][,STAY]

T HTUS,tid,pid,nid,UA=R[,DUPN=R][,DUPL=R][,STAY]
```

5.10.4 M (Macro) Subblocks

Macro Types:

GENLU ..... Lump generator
GENPA..... Path generator
GENT ..... Tie generator
LINE..... Generates a duct model with no ties
HX..... Generates a duct model with one row of ties
CAPPMP .... Capillary evaporator-pump model

Lump and Path Generator Subblock Formats:

```
M GENLU,mid,type,ilid[,N=I][,LUINC=I][,VOL=R]
      [,TL=R][,XL=R][,PL=R][,PL!=R][,VDOT=R][,QDOT=R]
      [,TLINC=R][,PLINC=R][,XLINC=R][,VOLINC=R]
      [,QDINC=R][,VDINC=R][,COMP=R][,CX=R][,CXINC=R]
      [,CY=R][,CYINC=R][,CZ=R][,CZINC=R]

M GENPA,mid,type,ipid,illid,il2id[,N=I][,PAINC=I]
      [,L1INC=I][,L2INC=I]
      [,FR=R][,FRINC=R][,STAT=C][,TWIN=I]
      [,DUPI=R][,DUPJ=R][,DUP=R]
      further path specifications ...
```

Generic Tie Generator Subblock Format:

```
M GENT,mid,type,itid,ilpid,inid {,fs}{,ipid,f1,ip2id,f2}
      [,N=I][,LUINC=I][,PAINC=I][,PA2INC=I][,NINC=I]
      [,TIINC=I]{,UA=R}[,DUPN=R][,DUPL=R][,STAY]
```

Type–Specific Tie Generator Subblock Formats:

```
M GENT,mid,HTU,itid,ilid,inid, UA = R
      [,N=I][,LUINC=I][,NINC=I]
      [,TIINC=I][,DUPN=R][,DUPL=R]

M GENT,mid,HTUS,itid,ipid,inid, UA = R
      [,N=I][,PAINC=I][,NINC=I]
      [,TIINC=I][,DUPN=R][,DUPL=R][,STAY]

M GENT,mid,HTN,itid,ilid,inid,ipid [,f1]
      [,N=I][,LUINC=I][,PAINC=I][,NINC=I]
      [,TIINC=I][,DUPN=R][,DUPL=R]

M GENT,mid,HTNS,itid,ipid,inid [,fs]
      [,N=I][,PAINC=I][,NINC=I]
      [,TIINC=I][,DUPN=R][,DUPL=R][,STAY]

M GENT,mid,HTNC,itid,ilid,inid,ipid,f1,ip2id [,f2]
      [,N=I][,LUINC=I][,PAINC=I][,PA2INC=I][,NINC=I]
      [,TIINC=I][,DUPN=R][,DUPL=R]

M GENT,mid,HXC,itid,ilid,inid,ipid [,f1]
      [,N=I][,LUINC=I][,PAINC=I][,NINC=I]
      [,TIINC=I][,DUPN=R][,DUPL=R]
```

Duct Generator Subblock Formats:

```
M LINE,mid,opt,ilid,ipid,l1id{,l2id}[,NSEG=I]
      [,TLENT=R][,DHS=R][,AFS=R]
      [,AFTH=R][,FK=R][,FR=R]
      [,LU=C][,PA=C]
      [,LUINC=I][,TL=R][,PL(!)=R][,XL=R][,QDOT=R]
      [,TLINC=R][,PLINC=R][,XLINC=R][,QDINC=R]
      [,PAINC=I][,WRF=R][,HC=R][,FC=R][,FPOW=R][,AC=R]
      [,IPDC=I][,UPF=R][,COMP=R][,CX=R][,CXINC=R]
      [,CY=R][,CYINC=R][,CZ=R][,CZINC=R]
      [,TWIN=I][,AM=R][,FG=R][,FD=R]
      [,DUPI=R][,DUPJ=R][,DUP=R]

M HX,mid,opt,ilid,ipid,itid,inid,l1id{,l2id}[,NSEG=I]
      [,NINC=I][,TLENT=R][,DHS=R][,AFS=R]
      [,AFTH=R][,FK=R][,FR=R]
      [,LU=C][,PA=C]
      [,LUINC=I][,TL=R][,PL(!)=R][,XL=R]
      [,TLINC=R][,PLINC=R][,XLINC=R]
      [,PAINC=I][,WRF=R][,HC=R][,FC=R][,FPOW=R]
      [,AC=R][,IPDC=I][,UPF=R][,COMP=R][,CX=R]
      [,CXINC=R][,CY=R][,CYINC=R][,CZ=R][,CZINC=R]
      [,TWIN=I][,AM=R][,FG=R][,FD=R]
      [,DUPI=R][,DUPJ=R][,DUP=R][,DUPN=R][,DUPL=R]
      [,AFRACT=R][,TIINC=I][,STAY]
```

Component Model Subblock Formats:

```
M CAPPMP,mid,opt,ucid,jid,dcid,ulid,dlid,
      {,tid,tsmn.nid},RC=R,CFC=R[,XVH=R][,XVL=R]
      [,TL=R][,PL=R][,XL=R][,QDOT=R][,FR=R]
      [,DUPI=R][,DUPJ=R][,DUP=R][,DUPN=R][,DUPL=R]
      [,VAPOR=I]{,UA=R}
```

5.11 FPROP Data

Header Subblock Format:

```
HEADER FPROP DATA, Nnnn [,uid [,abszro] ]
     [,keyword=value][,keyword=value]
```

Array Subblock Format:

AT, X, t_1,v_1, t_2,v_2
... ,t_i,v_i, ... t_n,v_n

FPROP DATA Restrictions:

a. Fluid ID (Nnnn) must be from 6000 to 9999.
b. Local uid must be input in order to input local abszro.
c. All reference-style single-phase inputs use same TREF. For 7000 series fluids, all two-phase vapor properties use the same TREFG, all two-phase liquid properties use the same TREFL.
d. Array-style inputs take priority over reference-style.
e. Temperatures in array data must increase monotonically.
f. All input properties must be in consistent units and positive.

5.11.1 Perfect Gas (8000 Series)

Keywords in Header Subblock:

```
RGAS=R [,TREF=R]
[,TMIN=R][,TMAX=R]
{,K=R}   [,KTC=R]
{,CP=R}  [,CPTC=R]
{,V=R}   [,VTC=R]
```

Optional Array Subblocks:

```
{AT,K, ...}
{AT,CP, ...}
{AT,V, ...}
```

5.11.2 Incompressible Liquid (9000 Series)

Keywords in Header Subblock:

```
[,TREF=R]
[,TMIN=R] [,TMAX=R]
{,K=R}   [,KTC=R]
{,CP=R}  [,CPTC=R]
{,V=R}   [,VTC=R]
{,D=R}   [,DTC=R]
```

Optional Array Subblocks:

```
{AT,K, ...}
{AT,CP, ...}
{AT,V, ...}
{AT,D, ...}
```

5.11.3 Simplified Two-Phase (7000 Series)

Keywords in Header Subblock:

```
HEADER FPROP DATA, 7nnn, [,uid [,abszro] ]
     RGAS=R, TCRIT=R, PCRIT=R [,TREFG=R] [,TREFL=R]
     {,KG=R} [,KGTC=R]          {,KL=R} [,KLTC=R]
     {,VG=R} [,VGTC=R]          {,VL=R} [,VLTC=R]
     {,CPG=R} [,CPGTC=R]        {,DL=R} [,DLTC=R]
     {,ST=R} [,STTC=R]
     [,AVDW=R] [,BVDW=R]
     [,TMIN=R] [,TGMAX=R] [,PGMAX=R]
     [,WSRK=R]
```

Required Array Subblock:

```
AT,DOME,t1,p1,hfg1,t2,p2,hfg2
```

Optional Array Subblocks:

```
{AT,KG, ...}
{AT,KL, ...}
{AT,VG, ...}
{AT,VL, ...}
{AT,CPG, ...}
{AT,DL, ...}
{AT,ST, ...}
```

5.11.4 Advanced Two-Phase (6000 Series)

Keywords in Header Subblock:

```
HEADER FPROP DATA, 6nnn, [uid [,abszro] ]
     TCRIT=R, PCRIT=R
     [,TMIN=R] [,TGMAX=R] [,PGMAX=R]
     [NEVERLIQ]
```

Reserved Names in Functional Subblocks:

T ..... Temperature, local units as selected on the HEADER card
P ..... Pressure, *absolute* local units, *double precision*
H ..... Enthalpy, local units
S ..... Entropy, local units
D ..... Density, local units
V ..... Specific volume, local units
CP ..... Specific heat, local units
U ..... Internal energy, local units
HFG ..... Heat of vaporization, local units
X ..... Quality (vapor mass fraction)
A ..... Void fraction (vapor volume fraction)
VISC ..... Viscosity, local units
COND ..... Conductivity, local units
SOS ..... Speed of sound, local units
METH ..... Two–phase speed of sound method: 1 or 2; *integer*
ST ..... Surface tension, local units
DPDT ..... Derivative of sat. pressure w.r.t. temperature: (dP/dT)sat
DPARG ..... RESERVED
SPARG ..... RESERVED
INTARG ..... RESERVED

Optional First Functional Subblock:

VINIT

Required Functional Subblocks (always):

VSV
VH
VVISCV
VCONDV

Required Functional Subblocks (if NEVERLIQ is not employed):

VPS
VDPDTT
VDL
VVISCF
VCONDF
VST

Optional Functional Subblocks, Additional Properties:

VS
VSOS
VTAV1
VQUALS
VDLC

Optional Functional Subblocks, More Appropriate Methods:

VTAV2
VTS
VDPDT
VCPV
VCPF
VHLIQ
VTLIQ
VULIQ
VQUALH
VHFG
VALPHA
VSOSF
VSOSFV

6. SUBROUTINE LIBRARY

This section describes the subroutines that may be called from within the logic blocks. Subroutines are available to perform network solutions, network mapping and output, restart and parametric operations, component simulations, mathematical operations, etc. Because the operations performed by the program are determined by which routines are called and in which order, it is well worth the reader's time to be familiar with all of the support routines available to him.

The following paragraphs describe the conventions that will be used in this section.

Data Types—Unless otherwise specified, formal arguments whose first letter is I, J, K, L, M, or N represent actual arguments which must be *integers*. Formal arguments beginning with any other letter represent actual arguments which must be *floating point* values, unless otherwise specified. When an actual argument must be a *character* value, this will be identified.

Reference Forms—Arguments which serve as input data to a subroutine may be supplied as literal data values or as variables. In the former case, the value must be of the type (i.e., integer, floating point, etc.) shown in the calling sequence. In the latter case, the variable name must refer to a memory location which contains a value of the proper type. Arguments which will receive the results (i.e., output) of a subroutine must be supplied as Fortran variables. Overlooking this restriction will lead to havoc.

Formal arguments that are suffixed with (IC) (e.g., smn.AX(IC), smn.A(IC), Y(IC), etc., smn refers to a submodel name) indicate that the actual argument supplied by the user *must be an array reference of the integer count form*. Integer count form is the convention for storage of array data in SINDA/FLUINT, where the first cell contains the size of the array stored as an integer. Formal arguments that are suffixed with (DV) (e.g., smn.G(DV), Q(DV), smn.T(DV), etc.) indicate that the actual argument supplied by the user must refer to the location of the first data value (i.e., the starting location) in an array. Hence, the user may supply an array reference of the *data value* (A(n+i) or smn.A(n+i)) form, or in general, any reference which refers to the starting location of the desired sequence of data values.

Actual To Relative Conversions—Some subroutine arguments rely upon the preprocessor's ability to convert from actual constant, array, node, conductor, lump, tie, and path numbers to relative numbers. To use this capability the user may supply an actual constant number, array number, node number, conductor number, lump number, tie number, or path number by preceding the actual number with K, A, T, G, or FLUINT variable respectively. This causes the preprocessor to replace the entry with the relative number. Consider for example the argument list shown below:

```
(2, A14, POD2.T5, G7)
```

In this example, following the preprocessor phase, A14 would be replaced by the pointer to the first cell in the A array, corresponding to the start of array number 14 (the user should remember that this first cell is equivalenced to the integer size of the array), T5 would be replaced by the relative node number for actual node number 5, and G7 would be replaced by the relative conductor number for actual conductor number 7.

Recall that all subroutine calls with arguments that require conversion must be made using M-type statements since F-type statements by-pass the conversion process. M-type statements begin with a blank, an M or an integer number in Column 1. F-type statements must begin with an F, or be bracketed by FSTART/FSTOP instructions. The majority of the library subroutines may be called with an M-type statement, whether or not conversions must be performed. In this section, some calling sequences will include an M in column 1 to emphasize that argument list conversion is required—an F is *not* allowed. In the few instances where an F is mandatory, it will be included in the calling sequence.

6.1 Subroutine Name Index

Name	Paragraph	Page	Description
ACKERS	6.9.3	6–112	Condensation Heat Transfer Coefficient
ADARIN	6.8.12	6–85	Calculates Inverse Sum of Inverse of Array
ADDARY	6.8.1	6–80	Adds Elements of Two Arrays
ADDMOD	6.6.11	6–66	Returns Therm. Submodel to Active Status
ARINDV	6.8.11	6–85	Divides Constant by Array
ARYADD	6.8.2	6–80	Adds Constant to Array
ARYDIV	6.8.4	6–81	Divides Array by Constant
ARYINV	6.8.10	6–84	Inverts Each Element of an Array
ARYMPY	6.8.6	6–82	Multiplies Array by Constant
ARYSUB	6.8.8	6–83	Subtracts Constant from Array
ARYTRN	6.6.1	6–61	Returns Absolute Location of an Array
BAROC	6.9.4	6–117	Two Phase Friction Gradient
BELACC	6.9.6.2	6–131	Simulates Two–phase Bellows Accumulator
BNDDRV	6.5.1	6–57	Drives Boundary Temperatures
BNDGET	6.4.11	6–52	Writes to ASCII File From SAVE File
CHENNB	6.9.3	6–114	Nucleate Boiling Heat Transfer Coef.
CHGLMP	6.9.1	6–89	Changes Lump Thermodynamic State
CHGSUC	6.9.1	6–91	Changes Path Phase Suction Status
CHGVOL	6.9.1	6–90	Changes Tank Volumes
CHKCHL	6.9.6.6	6–152	Assists user in simulating choked flow
CHKCHP	6.9.6.6	6–152	Monitors one or more paths for choking limit
CHOKER	6.9.6.6	6–152	Choked flow simulation for connectors
COMBAL	6.6.12	6–67	Calculates Energy Balance - Transient
COMPLQ	6.9.6.7	6–166	Adds liquid compressibility as compliance
COMPRS	6.9.6.1	6–129	Simulates Compressors and Turbines
CONTRN	6.6.2	6–62	Gives Relative Conductor Location
CPRINT	6.4.1	6–40	Prints Capacitance Values
CRASH	6.4.3	6–43	Creates and updates abort recovery file
CRVINT	6.6.3	6–62	Integrates Doublet Array
CRYTRN	6.6.4	6–63	Returns Index of UCA Array
CSIFLX	6.3.4	6–29	Cyclic Flux Interpolation
CVTEMP	6.3.1	6–22	Update Time-Dependent Capacitances
D11CYL	6.3.4	6–30	Linear Cyclic Interpolation - Doublet Array
D11DAI	6.3.3	6–26	Same as D1DG1I - Singlet Arrays
D11DIM	6.3.3	6–26	Same as D1D1IM - Singlet Arrays
D11MCY	6.3.4	6–30	Same as D11CYL With Multiplier Z
D11MDA	6.3.3	6–24	Linear Interpolation With Multiplier
D11MDI	6.3.3	6–26	Same as D1D1MI - Singlet Arrays
D11MDT	6.3.5	6–32	Linear Interpolation From TARRAY Data
D12CYL	6.3.4	6–30	Same as D11CYL - Parabolic Interpolation
D12MCY	6.3.4	6–31	Same as D11MCY - Parabolic Interpolation
D12MDA	6.3.6	6–33	Parabolic Interpolation With Multiplier Z
D1D1DA	6.3.3	6–24	Linear Interpolation on Singlet Array
D1D1IM	6.3.3	6–26	Same as D1DG1I With Constant Multiplier
D1D1MI	6.3.3	6–26	Linear Interpolation With Varying Multiplier
D1D1WM	6.3.3	6–25	Linear Interpolation With Constant Multiplier
D1D2DA	6.3.6	6–33	Parabolic Interpolation On Singlet Arrays
D1D2WM	6.3.6	6–33	Parabolic Interpolation With Multiplier
D1DEG1	6.3.3	6–24	Linear Interpolation On Doublet Arrays
D1DEG2	6.3.6	6–33	Parabolic Interpolation On Doublet Array
D1DG1I	6.3.3	6–26	Linear Interpolation On Array, Forms New

Name	Paragraph	Page	Description
D1IMD1	6.3.3	6-28	Linear Interp. Using Mean of Ind. Var.
D1IMIM	6.3.3	6-28	Same as D1IMD1 With Varying Multiplier
D1IMWM	6.3.3	6-28	Same as D1IMD1 With Constant Multiplier
D1M1DA	6.3.3	6-25	Linear Interpolation Using Mean Ind. Values
D1M1MD	6.3.3	6-25	Same as D1M1WM With Constant Multiplier
D1M1WM	6.3.3	6-25	Linear Interpolation Using Mean Ind. Var.
D1M2DA	6.3.6	6-33	Parabolic Interp. Using Mean Ind. Var.
D1M2MD	6.3.6	6-34	Same as D1M2WM With Constant Multiplier
D1M2WM	6.3.6	6-34	Parabolic Interp. Using Mean Ind. Values
D1MDG1	6.3.3	6-25	Linear Interpolation Using Mean Ind. Var.
D1MDG2	6.3.6	6-33	Parabolic Interp. Using Mean Ind. Var.
D2D1WM	6.3.3	6-25	Bivariate Linear Interpolation With Multiplier
D2DEG1	6.3.3	6-26	Bivariate Linear Interpolation
D2DEG2	6.3.6	6-34	Bivariate Parabolic Interpolation
D2D2WM	6.3.6	6-34	Bivariate Parabolic Interpolation With Mult.
D2MX1M	6.3.3	6-27	Biv. Linear Int. Mean Ind. Var. With Mult.
D2MX2M	6.3.6	6-34	Bivariate Parabolic Interpolation With Mult.
D2MXD1	6.3.3	6-27	Bivariate Linear Int. Using mean Ind. Var.
D2MXD2	6.3.6	6-35	Biv. Parabolic Int. Using Mean Ind. Var.
D3D1WM	6.3.3	6-24	Trivariate Linear Interpolation With Mult.
D3DEG1	6.3.3	6-24	Trivariate Linear Interpolation
DA11CY	6.3.4	6-30	Cyclic Linear Int. Singlet Arrays
DA11MC	6.3.4	6-29	Cyclic Linear Int. Singlet Arrays With Mult.
DA12CY	6.3.4	6-30	Cyclic Par. Int. Singlet Arrays
DA12MC	6.3.4	6-31	Cyclic Par. Int. Singlet Arrays With Mult.
DITTUS	6.9.3	6-111	Single-Phase Heat Transfer Coefficient
DIVARY	6.8.3	6-81	Divides One Array by Other Array
DRPMOD	6.6.10	6-66	Makes Therm. Submodel Inactive
D1D1DA	6.3.3	6-24	Linear Int. Using Singlet Arrays
DT11MC	6.3.5	6-32	Cyclic Linear Int. From TARRAY Data
FASTIC	6.2.5	6-18	Fast Initial Conditions (Steady-State Soln.)
FLOMAP	6.9.5	6-125	Prints Fluid Submodel Map
FORWRD	6.2.1	6-8	Explicit Forward Differencing
FRICT	6.9.4	6-115	Darcy Friction Factor
FRIEDL	6.9.4	6-117	Two Phase Friction Gradient
FRPRNT	6.9.5	6-124	Prints Flow Rates For Paths
FWDBCK	6.2.2	6-12	Forward-Backward Differencing
FWDMTS	6.2.3	6-15	FORWRD - Multiple Time Step
GADDi	6.7.5	6-77	Calculate Effective Network Conductance
GENOUT	6.4.8	6-49	Prints Out Arrays
GOHOMO	6.9.1	6-92	Forces Twinned Paths to Stay Homogeneous
GOSLIP	6.9.1	6-93	Undoes Actions of GOHOMO; Enables Slip Flow
GPRINT	6.4.1	6-40	Prints Out Conductances
GVTEMP	6.3.1	6-22	Update Temp.-Dependent Conductances
GVTIME	6.3.2	6-23	Update Time-Dependent Conductances
HEATER	6.7.3	6-71	Heater Switching
HLDLMP	6.9.1	6-94	Makes Tank or Junction Act like Plenum
HNQCAL	6.7.1	6-68	Heater Node Energy Requirements
HNQPNT	6.4.12	6-54	Prints Q's For Heater Nodes
HTRLMP	6.9.1	6-95	Hold Enthalpy of Lump

Name	Paragraph	Page	Description
HXCNT	6.7.4	6-72	Counter Flow Heat Exch. Effectiveness
HXCOND	6.7.4	6-73	Condensing Heat Exchanger
HXCROS	6.7.4	6-74	Cross Flow Heat Exchanger Effectiveness
HXEFF	6.7.4	6-72	Heat Exchanger Simulation
HXPAR	6.7.4	6-75	Parallel Flow Heat Exch. Effectiveness
INTLMP	6.9.1	6-97	Internal Pointer For a Given Lump
INTPAT	6.9.1	6-98	Internal Pointer For a Given Path
INTTIE	6.9.1	6-99	Internal Pointer For a Given Tie
LAGRAN	6.3.7	6-36	Lagrangian Int. Using Doublet Arrays
LGRNDA	6.3.7	6-36	Lagrangian Int. Using Singlet Arrays
LOCMAR	6.9.4	6-116	Two Phase Friction Gradient
LMPMAP	6.9.5	6-125	FLOMAP for One Lump
LMPRNT	6.9.5	6-123	Prints Lump Thermodynamic State Variables
LMPTAB	6.9.5	6-119	Tabulates Basic Lump Parameters
LMXTAB	6.9.5	6-120	Tabulates Extra Lump Parameters
LSTSQU	6.6.9	6-65	Least Square Curve Fit - Polynomial
MCADAM	6.9.4	6-116	Two Phase Friction Factor Gradient
MFLTRN	6.9.1	6-100	Relative Location of Fluid Submodel
MIXGAS	6.9.6.4	6-135	Simulates Mixing of Perfect Gases
MODTRN	6.6.5	6-63	Relative Location of Thermal Submodel
MPYARY	6.8.5	6-82	Multiplies Two Arrays
NCGBUB	6.9.6.3	6-133	Simulates Stationary Noncondensible Gas Bubble
NODMAP	6.4.2	6-41	QMAP for One Node
NODTRN	6.6.6	6-64	Relative Location of T, C, and Q For Node
NONEQ0	6.9.6.5	6-140	First Half of Noneq. Two-phase Simulation
NONEQ1	6.9.6.5	6-140	Last Half of Noneq. Two-phase Simulation
NONEQS	6.9.6.5	6-140	Set-up Options for Noneq. Two-phase Simulation
NUMTRN	6.6.7	6-64	Relative Location on UC of a Constant
PAGHED	6.4.14	6-55	Change page set-up (titles, line counts)
PRINT	6.4.13	6-54	Controls Printout Routine
PRINTA	6.4.8	6-49	Prints An Array Of Values
PRPMAP	6.9.5	6-126	Maps Working Fluid Properties
PTHTAB	6.9.5	6-121	Tabulates Path Parameters
PUTTIE	6.9.1	6-104	Moves a Tie to a New Lump and/or Node
QDPRNT	6.9.5	6-123	Prints Values of QDOT For Lumps
QFORCE	6.8.13	6-86	Multiplies A(1) by A(2)-A(3) Forms A(4)
QMAP	6.4.2	6-41	Prints Thermal Network Map
QMETER	6.8.14	6-86	Multiplies C1 by C2-C3 Forms C4
QMTRI	6.8.15	6-87	Multiplies A(1) by A(2)-A(3) Forms A(4)
QPRINT	6.4.1	6-40	Prints Nodal Heat Sources (Q)
QVTEMP	6.3.1	6-22	Update Temp.-Dependent Sources
QVTIME	6.3.2	6-23	Update Time-Dependent Sources
RELLMP	6.9.1	6-96	Releases HLDLMP and HTRLMP Actions
RESAVE	6.4.3	6-43	Stores Data For Restart
RESPAR	6.5.2	6-59	Re-Initializes For Parametric Cases
RESTAR	6.5.3	6-60	Reads In Parameters From Restart File
RESTNC	6.5.3	6-60	Same as RESTAR without Collision Checks
RDTNQS	6.8.16	6-87	[(T1)4 - (T2)4] * Constant
ROHSEN	6.9.3	6-112	Condensation Heat Transfer Coefficient

Name	Paragraph	Page	Description
SAVE	6.4.4	6-45	Writes Variables To SAVE File
SAVPAR	6.4.5	6-46	Saves Data For Parametric Cases
SHAH	6.9.3	6-113	Shah's Condensation Correlation
SORTPR	6.4.10	6-51	Prints Sorted Nodal Temperatures
SPRIME	6.9.1	6-102	Forces a Capillary Device to Prime
STDHTC	6.9.3	6-110	Convective Heat Transfer Coefficients
STDSTL	6.2.4	6-16	Steady State Solution
STNDRD	6.4.9	6-50	Prints Line of Data - Time, CSGMIN, etc.
STP1AM	6.3.8	6-38	Step Interpolation On Doublet Array
STP1AS	6.3.8	6-37	Step Interpolation On Doublet Array
STP2AM	6.3.8	6-38	Step Interpolation On Singlet Arrays
STP2AS	6.3.8	6-37	Step Interpolation On Singlet Arrays
STPRNT	6.9.5	6-124	Prints STUBE Parameters
SUBARY	6.8.7	6-83	Subtracts Two Arrays
SUMARY	6.8.9	6-84	Sums Elements of an Array
TBPRNT	6.9.5	6-124	Prints Tube Parameters
TDUMP	6.4.1	6-40	Prints Nodal Temperatures
THRMST	6.7.2	6-69	Thermostat Switch
TIETAB	6.9.5	6-120	Tabulates Tie Parameters
TKPRNT	6.9.5	6-123	Prints Tank Parameters
TMNMX	6.4.7	6-48	Prints Max./Min. Nodal Temperatures
TPRINT	6.4.1	6-40	Prints Nodal Temperatures
TRPZDA	6.6.8	6-65	Integrates By Trapezoidal Rule
TSAVE	6.4.6	6-47	Saves Temperatures For Post Processing
TUBTAB	6.9.5	6-122	Tabulates Tube and STUBE parameters
TWNTAB	6.9.5	6-121	Tabulates Twinned Tube and STUBE parameters
UDFERR	6.9.5	6-126	Error Routine for 6NNN Fluids
USRFIL	6.4.15	6-56	Opens an Extra User File
VALPHA through VVISCV	6.9.2	6-106	Fluid Property Routines
WAVLIM	6.9.6.7	6-166	Max. time step for fast hydrodynamic transients
XTRACT	6.9.1	6-101	Effective Path Suction Quality
YSMPWK	6.9.5	6-127	YSMP workspace utilization reporting

6.2 Network Solution Routines

This section describes the five network solution routines available. These routines, called exclusively from OPERATIONS DATA, are the central drivers for most of SINDA/FLUINT. They are divided into two types: steady state and transient. Steady–state routines are normally used to find a single time–independent state for the currently built configuration. This state may be the end–product of the analysis, or it may be used for specifying initial conditions for subsequent transient analyses. Transient routines perform integrations of the network governing equations over time, resulting in a series of network states.

The five routines are:

Within all logic blocks except OPERATIONS DATA, the output variable NSOL can be used to determine which solution routine is currently active. This very useful variable identifies the current solution routine, which can be used to alter the instructions in user logic blocks (VARIABLES and FLOGIC) and in user subroutines:

NSOL	= 0	FASTIC
	= 1	STDSTL
	= 2	FWDBCK
	= 3	FORWRD
	= 4	FWDMTS

This variable can be referenced, but should not be changed. For example, user simulation logic is often different in steady states than it is in transients. A simple check can be used to distinguish between the two:

```
IF (NSOL .LE. 1) THEN
        (steady state logic)
ELSE
        (transient logic)
ENDIF
```

6.2.1 Network Solution-Transient/Explicit

Subroutine Name: FORWRD

Description: (See also Figure 3–16.) This subroutine performs a transient thermal analysis by the explicit forward differencing method and fluid analysis by implicit (backward) differencing (see Appendix E for fluid network methods). First, a heat balance equation is written about a node in forward finite difference form:

$$\frac{C_i}{\Delta t}\left(T_i^{n+1} - T_i^n\right) - Q_i + \sum_{j=1}^{N}\left[G_{ji}\left(T_j^n - T_i^n\right) + \hat{G}_{ji}\left\{\left(T_j^n\right)^4 - \left(T_i^n\right)^4\right\}\right]$$

where:

T_j^n ..... temperature of node j at the current time t

T_i^{n+1} ..... temperature of node i at the next time t + Δt

G_{ji} ..... linear conductor attaching node j to node i (e.g., KA/L)

$\hat{G}_{ji}$ ..... radiation conductor attaching node j to node i (e.g., $\sigma A_i \mathcal{F}_{ji}$)

C_i ..... thermal capacitance of node i (e.g., $V \bullet \rho \bullet C_p$)

Q_i ..... source/sink for node i

This equation is the defining equation for diffusion nodes. As this equation is solved for node i, the heat flowing through the attached conductors is evaluated and used to adjust the source/sink on node j. (This heat is neglected for any one-way conductors in which node i is the upstream node.) The finite difference equation presented above uses only the temperature derivatives at the current time to predict the overall temperature change. This method is also called *Eulerian,* and is first-order accurate in time and space.

Because this equation can be solved explicitly for T_i^{n+1}, there is a limitation imposed upon the maximum time step due to stability considerations: the upper bound is the smallest of the ratios of the thermal capacitance to the sum of the linearized conductors (C/ΣG) for every diffusion node in the network. The term that is added in the sum of conductor for a radiation conductor to linearize it is:

$$\sigma A_i \mathcal{F}_{ij}(T_i + T_j)2(T_i + T_j)$$

For time steps in excess of this minimum ratio (called CSGMIN), the network temperatures may oscillate or diverge without limit. To prevent this, the routine computes CSGMIN and bases the time step on it. As the time step deviates from an infinitesimal quantity, so *might* the error imparted to the resulting temperatures. *Note that stability and accuracy are not related,*

and that keeping the time step less than CSGMIN guarantees stability but not accuracy. In most practical thermal applications, this time step criterion has been found to be adequate when used in combination with limits on the temperature change per time step (governed by DTMPCA as explained below).

Another point to remember is that since the heat balance equation was formulated using the temperatures of nodes i and j at the start of the time step (t), those boundary conditions that are functions of time should use the old time (TIMEO) for consistency.

The VARIABLES 1 and VARIABLES 0 operations are performed once each time step before the diffusion node temperature calculations and the VARIABLES 2 operations are performed after the relaxation of the arithmetic nodes at the end of the time step. The OUTPUT CALLS are performed in increments of OUTPUT starting from the input value of TIME0.

The time step is computed first as DTIMEU = CSGMIN * 0.95/CSGFAC. (If fluid submodels are active, a maximum value of DTIMEU has already been calculated for this interval.) The value of CSGMIN used is that computed from the last integration step. A check is made after the calculation of the last diffusion node to determine if the time step used (DTIMEU) is less than the time step just computed. If not, then DTIMEU is adjusted downward and the diffusion node temperatures are recalculated. The subroutine ensures that CSGFAC .GE. 1.0. The user may input a larger value if he desired. DTIMEU is set to the smallest of the thermal and fluid (DTIMUF) time steps if fluid submodels are active.

In addition, a "look ahead" feature has been incorporated in the routine to alter the value of DTIMEU. If one time step would exceed the output time point, the time step is adjusted so it ends exactly on the output time point. In addition, the user may explicitly control how large or how small DTIMEU may be by using DTIMEH and DTIMEL. The routine checks to see if DTIMEU .GT. DTIMEH; if so, then DTIMEU is set equal to DTIMEH; if DTIMEU .LT. DTIMEL, an error message is printed and the problem terminates after calling OUTPUT CALLS. If no input values are specified, DTIMEH is set to infinity and DTIMEL is set to zero.

The user may also explicitly control the maximum diffusion node temperature change over a time step (and implicitly control the time step). The largest temperature change for the set of diffusion nodes is stored in DTMPCC with the correct sign. A check is made to insure that |DTMPCC| .LT. DTMPCA. If not, DTIMEU is adjusted downward and the diffusion node temperatures are recalculated (without repeating fluid calculations). This check is always performed at the end of the diffusion node calculations. If no input value is specified, DTMPCA is set to 1.0E30.

After the diffusion node temperatures are calculated, the set of arithmetic nodes is then iterated to a steady-state solution. The diffusion node temperatures just calculated are then considered as boundary conditions for the relaxation of this set of arithmetic nodes. The arithmetic nodes are, by definition, steady-state nodes. The heat balance equation for node i on the $k+1^{st}$ iteration is:

$$0 = Q_i + \sum_{j=1}^{L} \left[G_{ij} \left(T_j^{k+1} - T_i^{k+1} \right) + \hat{G}_{ij} \left\{ \left(T_j^{k+1} \right)^4 - \left(T_i^{k+1} \right)^4 \right\} \right]$$

$$+ \sum_{j=1}^{N} \left[G_{ij} \left(T_j^{k} - T_i^{k+1} \right) + \hat{G}_{ij} \left\{ \left(T_j^{k} \right)^4 - \left(T_i^{k+1} \right)^4 \right\} \right]$$

where the superscript refers to the iteration.

This gives rise to a set of simultaneous equations which are solved in an iterative manner. The user must specify the convergence criterion (ARLXCA) for the solution and the maximum number of iterations (NLOOPT) to be performed. The largest temperature change from one iteration to the next is stored in ARLXCC during each iteration. Iterations will be performed until |ARLXCC| .LT. ARLXCA or until the number of iterations (LOOPCT) is greater than NLOOPT. If there are arithmetic nodes in the problem, both ARLXCA and NLOOPT must be specified. To accelerate the convergence, a variation of Aitken's delta-squared process is performed every third (or ITERXTth) iteration. This acceleration is only performed on those nodes that have exhibited a monotonic tendency over the previous three iterations, otherwise the acceleration actually works as a damping action. The user can control the change in the temperatures imparted by this acceleration process by use of EXTLIM. If LOOPCT = NLOOPT, a warning message is printed and the problem continues.

Once the arithmetic node temperatures have been calculated, the largest temperature change over the time step is stored in ATMPCC. A check is made to insure that |ATMPCC| .LT. ATMPCA; if not, DTIMEU is adjusted downward and both the diffusion and arithmetic node temperatures are recalculated. If no input value is specified, ATMPCA is set to 1.0E30. Note that the ATMPCA criterion can fail again on the second pass due to the fact that arithmetic node temperature changes can be time-independent. To avoid an infinite loop, the program will not attempt to reduce the time step twice in a row for this criterion, with the result that |ATMPCC| may occasionally be larger than ATMPCA. For this reason, the user should rely on DTMPCA rather than ATMPCA whenever possible.

The control constant BACKUP is tested for a non-zero real value after the calls to perform the VARIABLES 0, VARIABLES 1 and VARIABLES 2 operations. A non-zero real value in the former case will cause a recomputation of DTIMEU, TIMEN, and TIMEM. DTIMEU will again be compared against DTIMEH and DTIMEL and DTIMUF. BACKUP is then zeroed out, the nodal Q values are returned to their SOURCE DATA values, and the complete time step solution is performed again. The user's logic should ensure that BACKUP is not perpetually set, and that some achievable criterion is used to allow the program to continue. At this point, the final fluid submodel calculations have not been completed; fluid submodels do not recognize the BACKUP command. Fluid calculations are completed upon successful completion of the thermal calculations using the final time step value.

The control constant OPEITR is tested for a non-zero value after the completion of the VARIABLES 2 operations. If non-zero it will cause the OUTPUT CALLS operations to be performed. OPEITR is thus used to cause output at time points other than the normal output time points.

Restrictions - Values must be assigned to control constants TIMEND and OUTPUT and/or OUTPTF. There must be at least one diffusion node in the network unless fluid networks are present, in which case at least one boundary node must be present. The problem start time, TIMEO, will be zero unless it has been reset by the user or another transient routine has been called earlier.

Calling Sequence:

```
CALL FORWRD
```

6.2.2 Network Solution-Transient/Implicit Second Order

Subroutine Name: FWDBCK

Description: (See also Figure 3–15.) This subroutine performs a transient thermal analysis by implicit forward-backward differencing and fluid analysis by implicit backward differencing (see Appendix E for a description of the fluid network methods). One heat balance equation is written about a diffusion node as a forward finite difference equation and another as a backward finite difference equation. The resulting sum of these two equations would be:

$$\frac{2C_i}{\Delta t}\left(T_i^{n+1} - T_i^n\right) - 2Q_i + \sum_{j=1}^{N}\left[G_{ji}\left(T_j^n - T_i^n\right) + \hat{G}_{ji}\left\{\left(T_j^n\right)^4 - \left(T_i^n\right)^4\right\}\right]$$
$$+ \sum_{j=1}^{N}\left[G_{ji}\left(T_j^{n+1} - T_i^{n+1}\right) + \hat{G}_{ji}\left\{\left(T_j^{n+1}\right)^4 - \left(T_i^{n+1}\right)^4\right\}\right]$$

where:

T_j^n ..... temperature of node j at the current time t

T_i^{n+1} ..... temperature of node i at the next time t + Δt

G_{ji} ..... linear conductor attaching node j to node i (e.g., KA/L)

$\hat{G}_{ji}$ ..... radiation conductor attaching node j to node i (e.g., $\sigma A_j \mathcal{F}_{ji}$)

C_i ..... thermal capacitance of node i (e.g., $V \bullet \rho \bullet C_p$)

Q_i ..... source/sink for node i

In effect, the finite difference equation presented above uses the average of the temperature derivatives at the current and next times to predict the overall temperature change. This method is also called *trapezoidal,* and is second order accurate in time and first order accurate in space. When applied to an orderly mesh of diffusion nodes and linear conductors, it represents the Crank-Nicholson approximation to the partial differential equation for conduction.

For an arithmetic node, the heat balance equation is, as always:

$$0 - Q_i + \sum_{j=1}^{I-1}\left[G_{ji}\left(T_j^{k+1} - T_i^{k+1}\right) + \hat{G}_{ji}\left\{\left(T_j^{k+1}\right)^4 - \left(T_i^{k+1}\right)^4\right\}\right]$$
$$+ \sum_{j=I}^{N}\left[G_{ji}\left(T_j^{k} - T_i^{k+1}\right) + \hat{G}_{ji}\left\{\left(T_j^{k}\right)^4 - \left(T_i^{k+1}\right)^4\right\}\right]$$

The preceding equations give rise to a system of equations that describe the network. These equations, being implicit in T^{n+1}, are solved by iterative relaxation. Unlike the explicit method, there is no upper bound on the time step *due to stability considerations* for this implicit method. However, the user must realize that as the time step becomes larger, so might the *error* imparted to the resulting temperatures.

Another point to remember is that since the heat balance equation was formulated using the temperatures of node i and node j at both the start (t) and the end of the time step ($t+\Delta t$), to remain consistent, those boundary conditions that are functions of time should use a mean value of these two time points (control constant TIMEM).

The VARIABLES 0 and VARIABLES 1 operations are performed once per time step. The VARIABLES 2 operations are performed after the iterations at the end of the time step. The OUTPUT CALLS operations are performed in intervals of OUTPUT starting from the input value of TIMEO.

The time step explicit in the calculation of the diffusion nodes is determined from the input value of DTIMEI or, if DTIMEI = 0.0, from an internal time-step calculation made after each time increment based on temperature trends. This time step, DTIMEU, is subject to the following limiting adjustments. DTIMEU is reset if it is larger than the smallest fluid submodel time step (which is used if there are no thermal submodels). DTIMEU may be adjusted by a "look ahead" feature; that is, if a time step would exceed the output time point, the time step is adjusted to come out exactly on the output time point. In addition, the user may place limits on how large or how small DTIMEU is by using DTIMEH and DTIMEL. Then, if DTIMEU is greater than DTIMEH, DTIMEU is set equal DTIMEH; if DTIMEU is less than DTIMEL, then an error message is printed and the problem terminates. If no input values are specified, DTIMEH is set to 1.0E30 and DTIMEL is set to zero.

The largest temperature change from one iteration to the next for the diffusion nodes is stored in DRLXCC; that for the arithmetic nodes is stored in ARLXCC. Relaxation iterations will continue to be performed until |DRLXCC| .LT. DRLXCA and |ARLXCC| .LT. ARLXCA. DRLXCA and ARLXCA are both defaulted to 0.01. Both criterion must be met before the relaxation is considered finished. If the iteration counter LOOPCT is greater than NLOOPT, a warning message is printed and that relaxation is considered complete.

To accelerate the convergence of the nodal temperatures, Aitken's del-squared process is used every ITERXTth iteration. This consists of determining the limit of the sequence of the last three successive temperatures for a node. This acceleration process is performed only on those nodes whose temperatures were monotonically increasing or decreasing over the previous three iterations, otherwise the acceleration actually performs damping. The user can limit the change in the temperatures imparted by this process by the use of EXTLIM.

After the relaxation is complete, the largest temperature changes over the time step are stored for each submodel. Those for the diffusion nodes are stored in DTMPCC, and those for the arithmetic nodes are stored in ATMPCC. A check is made to determine whether |DTMPCC| .LT. DTMPCA and |ATMPCC| .LT. ATMPCA are satisfied for each submodel. If not, DTIMEU is adjusted downward and the relaxation process is started from the beginning. If DTMPCA and/or ATMPCA are not specified, they will be set to 1.0E30. Note that the ATMPCA criterion can fail again on the second pass due to the fact that arithmetic node temperature

changes can be time-independent. To avoid an infinite loop, the program will not attempt to reduce the time step twice in a row for this criterion, with the result that |ATMPCC| may occasionally be larger than ATMPCA. For this reason, the user should rely on DTMPCA rather than ATMPCA whenever possible.

The VARIABLES 2 operations will be performed only after the successful check on the maximum temperature change over the time step.

After the calls to perform the VARIABLES 2 operations the value of control constant BACKUP is tested. If BACKUP is a non-zero real value after the call for the VARIABLES 2 operations, the calculated temperatures will be replaced with their values at the start of the time interval, and the time step solution will be repeated. The user's logic should ensure that BACKUP is not perpetually set, and that some achievable criterion is used to allow the program to continue. At this point, the final fluid submodel calculations have not been completed; fluid submodels do not recognize the BACKUP command. Fluid calculations are completed upon successful completion of the thermal calculations using the final time step value.

After the VARIABLES 2 operations, the value of control constant OPEITR is tested. If it is non-zero, the OUTPUT CALLS operations will be performed. OPEITR is thus used to cause output at each time point.

Restrictions: There must be at least one diffusion node in the network if no fluid submodels are present. Values *must* be assigned to control constants TIMEND and OUTPUT. There must be at least one diffusion node in the network unless fluid submodels are present, in which case at least one boundary node must be present. A problem may not start at a time other than zero unless control constant TIMEO is appropriately set.

Calling Sequence:

```
CALL FWDBCK
```

6.2.3 Explicit Differencing Using Multiple Time Steps

Subroutine Name: FWDMTS

Description: This subroutine is a variant of subroutine FORWRD that provides a split-integration (multiple time step) capability. FORWRD marches through a transient solution using a time step equal to 0.95 times the minimum CSGMIN of all thermal submodels. FWDMTS marches each submodel forward using a time step equal to 0.95 times the CSGMIN for the particular submodel. The subroutine operates such that the more massive (slower changing) portions of the model serve as temperature boundary conditions for the faster changing portions. This allows more efficient computation because computer resources are concentrated on the fast changing nodes that require the most computation steps. This split integration scheme works as follows: To begin, CSGMIN is computed for each submodel and each submodel is stepped forward one time step related to its individual CSGMIN. Thereafter, the order in which temperatures are updated for the various submodels is controlled by the TIMEN and CSGMIN for each submodel. The submodel with the smallest CSGMIN is stepped forward until its TIMEN is greater than or equal to that of the submodel with the next smallest CSGMIN. At this point, that submodel is advanced by its time step. The lowest CSGMIN submodel is then marched ahead until its TIMEN again exceeds some submodel's TIMEN, and the process is repeated. In the process of updating a submodel's temperatures, temperatures from the submodels in the boundary state are obtained for the point in time by linear interpolation. This is feasible because the temperatures for each boundary state submodel have already been stepped forward to a point beyond the current submodel's TIMEN.

Restrictions and guidance: *Fluid submodels are ignored with FWDMTS.* Otherwise the same restrictions and guidance stated for FORWRD apply for FWDMTS.

FWDMTS may give inaccurate results for submodels that are thermally dominated by energy flowing to or from other submodels through inter-submodel conductors. The criteria is used to measure this condition is based on the ratio CSGMOD/CSGMIN, where CSGMOD is the sum of a submodels capacitances divided by the sum of the conductors connecting the submodel to any arithmetic or diffusion node in any other submodel. CSGMIN is the usual minimum nodal capacitance divided by the sum of its conductors for all the diffusion nodes in the submodel. Testing has shown that if this ratio is greater than 100.0 for each submodel in the model, the submodels are sufficiently independent to qualify for FWDMTS. CSGMOD/CSGMIN is computed for each submodel and compared to 100. If any submodel fails this criteria, FORWRD is called automatically to provide an accurate solution.

The user is cautioned that this criteria test is only made once, using the conditions that exist when FWDMTS is called. If the model contains time or temperature dependent capacitances and/or conductors, FWDMTS should perhaps not be used because a CSGMOD/CSGMIN could change to the point where FWDMTS would begin to give inaccurate results.

Calling Sequence:

```
CALL FWDMTS
```

6.2.4 Network Solution-Steady State

Subroutine Name: STDSTL

Description: (See also Figure 3–17.) This subroutine calculates the steady-state solution of a fluid/thermal network (see Appendix E for the fluid solution methods, which are the same for STDSTL as they are for FORWRD and FWDBCK; see FASTIC for an alternate fluid steady state approach). The diffusion nodes and arithmetic nodes are solved by a "successive point" iterative method. The heat balance for the i^{th} node on the $k+1^{st}$ iteration is as follows:

$$0 = Q_i + \sum_{j=1}^{i-1} \left[G_{ij} \left(T_j^{k+1} - T_i^{k+1} \right) + \hat{G}_{ij} \left\{ \left(T_j^{k+1} \right)^4 - \left(T_i^{k+1} \right)^4 \right\} \right]$$

$$+ \sum_{j=i}^{N_i} \left[G_{ij} \left(T_j^{k} - T_i^{k+1} \right) + \hat{G}_{ij} \left\{ \left(T_j^{k} \right)^4 - \left(T_i^{k+1} \right)^4 \right\} \right]$$

where the superscript refers to the iteration count. The reader will recognize this equation as the equation used to find the temperatures of arithmetic nodes in transients. In STDSTL and FASTIC, diffusion nodes are treated like arithmetic nodes.

To accelerate the convergence of the solution, a variation of Aitken's del-squared process is used. This process consists of performing ITERXT iterations and then determining the limit of the resulting temperature sequence. If this nodal temperature sequence is not monotonically increasing or decreasing, the acceleration process actually results in damping. The user can control the change in the temperature imparted by this process by use of EXTLIM.

The user must specify the maximum number of iterations to be performed (control constant NLOOPS) in relaxing to steady state. He must also select the relaxation criteria that specify when the problem is at steady state. These criteria can consist of either the largest of the absolute values of the nodal temperature changes per iteration (DRLXCA for diffusion nodes and/or ARLXCA for arithmetic nodes), or the thermal energy balance values (EBALSA and/or EBALNA), or both. If the largest temperature change is used, the routine iterates until the criterion is met (or until both criteria are met if both types of nodes are present). The largest nodal temperature change for diffusion nodes is stored in DRLXCC; that for arithmetic nodes is stored in ARLXCC. The temperature change criterion is considered achieved when |DRLXCC| .LT. DRLXCA and |ARLXCC| .LT. ARLXCA for each submodel.

After the temperature change criteria have been satisfied, if the control constant EBALSA is zero for each submodel, the relaxation iterations are terminated and the VARIABLES 2 and OUTPUT CALLS operations are performed. If EBALSA is greater than zero, then more iterations will be performed until the relative error in all energy flows to and from the entire model is less than EBALSA. An analogous node-by-node check is performed for nonzero EBALNA based on absolute error in the energy flow for each node—EBALNA has units of power whereas EBALSA is unitless. Fluid submodels must also converge according to the RERRF and REBALF criteria. When all active criteria are met, a message is printed and the OUTPUT CALLS operations are performed.

Note that the temperature-change relaxation criterion must not be considered as a tolerance associated with the final temperature results: rather, it is nothing more than the maximum temperature change from one iteration to the next. Even though the temperature change per iteration tends to approach zero as the network approaches a steady-state condition, the magnitude of this number is not a sufficient condition for equilibrium. The necessary and sufficient condition for a steady state is that the energy entering each node equals the energy leaving that node, which implies that the energy flows at the network level are balanced.

STDSTL takes advantage of the thermal submodel organization of the pseudo compute sequence in order to reduce the number of calculations *(but not total iterations)* required to achieve steady state. When a particular submodel meets the convergence criteria (temperature change, energy balance or both) it is put in a dormant condition while iterations are performed on the remaining submodels. In the dormant state, a submodel's temperatures act as boundary temperatures for the remaining submodels. After ITHOLD iterations (default=10) in the dormant state, a dormant submodel is reactivated and it remains in the iteration loop until it again meets the convergence criteria. When the last submodel converges, all the dormant submodels are reactivated for at least one iteration loop in order to verify that convergence has indeed been achieved. Note that the use of nonzero ITHOLD may cause the final value of LOOPCT to appear higher than if ITHOLD were zero. This may give the illusion of having taken more computer time; the actual computer time has been reduced because of dormant submodels. In fact, this is the sole purpose of ITHOLD.

VARIABLES 0, VARIABLES 1 and OUTPUT CALLS operations are performed initially to provide initial conditions, and the VARIABLES 1 operations are performed every iteration until the solution is achieved. Upon meeting convergence criteria both the VARIABLES 2 and OUTPUT CALLS operations are performed. Control constant ITEROT is checked at the end of each iteration. If LOOPCT is non-zero and an integer multiple of ITEROT, then OUTPUT CALLS operations are performed. If the iteration count LOOPCT exceeds NLOOPS, a message is printed and VARIABLES 2 and OUTPUT CALLS operations are performed.

If a number of different steady-state solutions are required, the control constant TIMEND is used. TIMEN is incremented by DTIMES when a problem terminates whether it is due to achieving the relaxation criterion or to exceeding the maximum number of iterations. TIMEN can be used in the VARIABLES 0 or VARIABLES 1 blocks to make boundary conditions a function of this pseudo time. If TIMEND is greater than TIMEN, the routine will continue to perform steady-state solutions until TIMEN equals TIMEND.

If desired, the STDSTL call can be followed by a call to one of the transient solution subroutines. In this manner, the steady-state solution becomes the initial condition for the transient analysis. Because of this typical usage, if TIMEND is greater than TIMEO yet DTIMES is zero, the program will perform one steady state solution at TIMEO assuming that the value for TIMEND is intended for a later call to a transient routine.

Calling Sequence:

```
CALL STDSTL
```

6.2.5 Network Solution-Steady State/Fluid Initial Conditions

Subroutine Name: FASTIC

Description: (See also Figure 3–17.) This routine finds a steady-state solution for thermal and fluid networks. FASTIC performs exactly the same calculations for thermal networks as does STDSTL, so that discussion (Section 6.2.4) will not repeated here. Fluid calculations in FASTIC are fundamentally different than in any other solution routine. The fluid network methods for other routines are described in Appendix E.

As the name implies, FASTIC is intended primarily to find initial conditions for fluid submodels. Because of the primary FLUINT solution methods, rough initial conditions are equivalent to severe transient events which would require many time steps and iterations. FASTIC takes rough-guess initial conditions and "smooths" them into a consistent set. Subsequent calls to other solution routines (including STDSTL) can then concentrate on real transients. FASTIC can also be used as the primary solution routine.

FASTIC performs a successive point relaxation similar to the solution method in STDSTL for thermal networks. A simultaneous hydraulic (pressure/flowrate) solution is performed first in each iterative pass, followed by iterative energy (enthalpy/heat rate) solutions. Hydraulic solutions are always performed with a simultaneous one-pass method unless the matrix is ill-formed, and energy solutions are always completely iterative with two to three passes per iteration. If twinned paths are active, or if EI or EJ is nonzero for any path, then the simultaneous hydraulic solution becomes a coupled hydraulic and energy solution similar to that used in other solution routines.

In general, FASTIC initially approaches a steady-state solution in fewer iterations than STDSTL. Not all modeling options are possible with FASTIC, as noted below. Although most models are served best by either FASTIC or STDSTL, perhaps the best overall strategy is to call FASTIC and then STDSTL. All STDSTL control constants apply to FASTIC except RSMAXF and RSTUBF. RSSIZF is used only in a limited capacity. The defaults are the same for both routines.

Both FASTIC and STDSTL take cautious first steps or preliminary iterations. The purpose of these cautious steps is to absorb inconsistencies and begin to approach a solution monotonically without falsely exceeding the property range limits. Convergence is not allowed during the preliminary iterations. Because of this, two consecutive calls to steady-state routines will require at least 2 iterations in the second call to confirm convergence found in the first. FASTIC is often called prior to STDSTL. STDSTL usually converges immediately on a solution just achieved by FASTIC, but not always. The reasons for this apparent discrepancy include the fact that STDSTL supports all options whereas FASTIC does not, and that STDSTL performs a simultaneous energy balance that immediately sweeps out cumulative errors left by FASTIC iterative energy methods.

The routines HLDLMP, HTRLMP, and RELLMP can be used to control the results of FASTIC (as well as other solutions). By making a tank or junction act temporarily as a plenum or *reference lump* with HLDLMP, the user should be able to customize the solution by isolating parts of the network. For example, by holding the inlet and outlet of a subnetwork constant, the solution of the subnetwork becomes independent of the rest of the network. Similarly, HTRLMP can be used to hold the enthalpy of a lump.

Restrictions: During a FASTIC solution, tank states and tube flowrates may change instantly—*in every way, tanks are treated like junctions and tubes are treated like STUBE connectors*. This imposes certain restrictions in program usage. First, VDOT and COMP terms are ignored. Second, at lease one lump must be either a plenum or a reference lump (state held by a previous call to HLDLMP). If the model contains no plena and no lump state has been held, a fatal error will result. Customizing QDOT values for tied lumps and tube/STUBE FC and FPOW values is not possible—changes to these parameters in FLOGIC 1 blocks are ignored. However, QDOT values for other lumps can be altered as with other solution routines.

Calling Sequence:

```
CALL FASTIC
```

6.3 Interpolation - Extrapolation Subroutines

Temperature-Dependent Variable Interpolation (6.3.1):

CVTEMP .... Linear interpolation of capacitance vs. temperature according to NODE DATA block inputs.
GVTEMP .... Linear interpolation of conductor values vs. temperature according to CONDUCTOR DATA block inputs.
QVTEMP .... Linear interpolation of Q-sources vs. temperature according to SOURCE DATA block inputs.

Time-Dependent Variable Interpolation (6.3.2):

GVTIME .... Linear interpolation of conductor values vs. time according to CONDUCTOR DATA block inputs.
QVTIME .... Linear interpolation of Q-sources vs. time according to SOURCE DATA block inputs.

Generalized Linear Interpolation (6.3.3):

D1DEG1 .... Linear interpolation using doublet array.
D1D1DA .... Linear interpolation using two singlet arrays.
D11MDA .... Same as D1D1DA except allows single value multiplier.
D3DEG1 .... Linear interpolation using trivariate arrays.
D3D1WM .... Trivariate linear interpolation with single value.
D1D1WM .... Uses D1DEG1 and multiplies the interpolation by the Z value.
D2D1WM .... Bivariate liner interpolation with single value multiplier.
D1MDG1 .... Same as D1DEG1 except arithmetic mean of two input values used as independent variable.
D1M1DA .... Same as D1MDG1 except uses two singlet arrays.
D1M1WM ... Same as D1MDG1 except a single value multiplier is allowed.
D1M1MD .... Same as D1M1DA except a single value multiplier is allowed.
D1DG1I ..... Perform interpolation on an array of X's to obtain an array of Y's.
D1D1IM ..... Same as D1DG1I with a single value multiplier.
D1D1MI ..... Same as D1D1IM except an array of multipliers is allowed.
D11DAI ..... Same as D1DG1I except uses two singlet arrays.
D11DIM ..... Same as D1D1IM except uses two singlet arrays.
D11MDI ..... Same as D1D1MI except uses two singlet arrays.
D2DEG1 .... Linear interpolation using bivariate arrays.
D2MXD1 .... Same as D2DEG1 except uses arithmetic mean of two values as independent variable.
D2MX1M .... Same as D2MXD1 except a single value multiplier is allowed.
D1IMD1 ..... Double array linear interpolation using arithmetic mean of two values as independent variable.
D1IMWM .... Same as D1IMD1 with a single value multiplier.
D1IMIM ..... Same as D1IMD1 except allows an array of multipliers.

Cyclical Interpolation (6.3.4):

DA11MC .... Same as DA11CY except allows single value multiplier.
CSIFLX ..... Same as DA11MC except there are two dependent variables for each independent variable
D11MCY .... Same as D11CYL except allows single value multiplier.
D11CYL ..... Cyclical linear interpolation using a single cycle doublet array.
DA11CY ..... Same as D11CYL except two singlet arrays are used.
D12CYL ..... Same as D11CYL except a parabolic interpolation is performed.
DA12CY ..... Same as DA11CY except a parabolic interpolation is performed.
D12MCY .... Same as D11MCY except parabolic interpolation is performed.
DA12MC .... Same as DA11MC except parabolic interpolation is performed.

Interpolation using TARRAY Data (6.3.5):

D11MDT .... Same as D11MDA except uses TARRAY data to save core.
DT11MC .... Same as DA11MC except uses TARRAY data to save core.

Generalized Parabolic Interpolation (6.3.6):

D1DEG2 .... Parabolic interpolation using a doublet array.
D1D2DA .... Parabolic interpolation using two singlet arrays.
D1D2WM .... Same as D2DEG2 except a single value multiplier is allowed.
D12MDA .... Same as D1D2WM except uses two singlet arrays.
D1MDG2 .... Same as D1DEG2 except the arithmetic average of two values is used as independent variable.
D1M2DA .... Same as D1D2DA except the arithmetic average of two values is used as independent variable.
D1M2WM ... Same as D1MDG2 except a single value multiplier is used.
D1M2MD .... Same as D1M2WM except uses two singlet arrays.
D2DEG2 .... Parabolic interpolation using a bivariate array.
D2D2WM .... Same as D2DEG2 except a single value multiplier is used.
D2MX2M .... Same as D2DEG2 except the arithmetic mean of two input values is used as independent variable.
D2MXD2 .... Same as D2D2WM except the arithmetic mean of two input values is used as independent variable.

Generalized Lagrangian Interpolation (6.3.7):

LAGRAN .... Performs Lagrangian interpolation using a doublet array.
LGRNDA .... Performs Lagrangian interpolation using two singlet arrays.

Step Interpolation (6.3.8):

STP1AS ..... Performs step interpolation using doublet arrays.
STP2AS ..... Performs step interpolation using two singlet arrays.
STP1AM .... Performs step interpolation on doublet arrays using a singlet array if X's.
STP2AM .... Same as STP1AM except uses singlet X and singlet Y arrays.

6.3.1 Temperature-Dependent Variable Interpolation

Subroutine Names: CVTEMP, GVTEMP, QVTEMP

Description: These subroutines perform the capacitance, conductance and Q-source vs. temperature interpolations required to support the temperature-dependent options allowed in the NODE, CONDUCTOR and SOURCE DATA blocks. They also account for FAC records and user constant multipliers used in the temperature-dependent CAPACITANCE, SOURCE and CONDUCTORS DATA inputs. They all interpolate on arrays input in the ARRAY DATA blocks. Linear interpolation with Z value multiplication on the result is performed. In multi-submodel problems a subroutine call is required to update each submodel's parameters.

Restrictions and Guidance: Calls to these routines at the end of each VARIABLES 1 block are generated by the preprocessor. The user may insert VARIABLES 1 statements after the calls by inserting his own calls to CVTEMP, GVTEMP and/or QVTEMP. User calls to these routines may be appropriate in OPERATIONS DATA for initialization. Calls from VARIABLES 0, VARIABLES 2 and OUTPUT CALLS are inappropriate. User calls to these routines *must not* have an "F" in column 1.

Calling Sequences:

```
M       CALL CVTEMP ('SMN')
M       CALL GVTEMP ('SMN')
M       CALL QVTEMP ('SMN')
```

where SMN is a thermal submodel name.

6.3.2 Time-Dependent Variable Interpolation

Subroutine Names: GVTIME, QVTIME

Description: These subroutines perform the conductor and Q-source interpolations required to support the time-dependent options allowed in the CONDUCTOR and SOURCE DATA blocks. They also account for FAC records and user constant multipliers that are used in the time-dependent SOURCE and CONDUCTOR DATA inputs. They interpolate on dual singlet arrays input in either the ARRAY DATA or TARRAY DATA blocks. Linear interpolation with Z-value multiplication on the result is performed. In multi-submodel problems, a subroutine call is required for updating each submodel's parameters.

Restrictions and Guidance: Calls to these routines at the end of the VARIABLES 0 blocks are generated by the preprocessor. This means that the CONDUCTOR DATA and SOURCE DATA blocks override any user-input VARIABLES 0 code as a default condition. To override this, the user may insert a call QVTIME (SMN) into VARIABLES 0, and follow this with any logic required to override source data. The same procedure also works for GVTIME.

User calls to these routines may be appropriate in OPERATIONS DATA for initialization. Calls from VARIABLES 1, VARIABLES 2 and OUTPUT CALLS are inappropriate. User calls to these routines *must not* have an "F" in column 1.

Calling Sequences:

```
M       CALL GVTIME ('SMN')
M       CALL QVTIME ('SMN')
```

where SMN is a submodel name.

6.3.3 Generalized Linear Interpolation

Subroutine Names: D1DEG1, D1D1DA, D11MDA

Description: Subroutines D1DEG1 and D1D1DA perform a linear interpolation using a doublet array and two singlet arrays respectively. Subroutine D11MDA uses D1D1DA and multiplies the result by a Z-value.

Restrictions and Guidance: These subroutines may be called from any logic block. All independent variable array values must be entered in monotonically increasing order. If the independent variable value supplied is outside the array's range, the appropriate end-value will be used. Z-value multipliers and X-values may be literal floating point numbers, user constants, program variables or elements from a user's array.

Calling Sequences:

```
M       CALL D1DEG1 (X,A(IC),Y)
M       CALL D1D1DA (X,AX(IC),AY(IC),Y)
M       CALL D11MDA (X,AX(IC),AY(IC),Z,Y)
```

where:

X	Value of independent variable
A	Doublet array of X-Y pairs
AX	Singlet array reference of the form smn.An or An
AY	Singlet array reference to array of Y values corresponding to each X value in AX (form smn.An or An)
Z	Value result is multiplied by before being returned as Y
Y	Dependent variable result

Subroutine Names: D3DEG1, D3D1WM

Description: These subroutines perform trivariate linear interpolation. The interpolation array must be constructed as shown for trivariate array format. (See Section 3.) Subroutine D2DEG1 is called on which calls on D1DEG1. Hence, the linear extrapolation feature of these routines applies. Subroutine D3D1WM multiplies the interpolated answer by F prior to returning it as T.

Restrictions: See trivariate array format. F must be a floating point value.

Calling Sequence:

```
M       CALL D3DEG1(X,Y,Z,TA(IC),T)
M       CALL D3D1WM(X,Y,Z,TA(IC),F,T)
```

Subroutine Names: D1D1WM

Description: This subroutine performs single variable linear interpolation by calling on D1DEG1. However, the interpolated answer is multiplied by the values addressed as Z prior to being returned as Y.

Restrictions: Same as D1DEG1 and Z must be a floating point number.

Calling Sequence:

```
M       CALL D1D1WM(X,A(IC),Z,Y)
```

Subroutine Name: D2D1WM

Description: This subroutine performs bivariate linear interpolation by calling on D2DEG1. The interpolated answer is multiplied by the W value prior to being returned as Z.

Restrictions: Same as D2DEG1 and W must be a floating point value.

Calling Sequence:

```
M       CALL D2D1WM (X, Y, BA(IC), W, Z)
```

Subroutine Names: D1MDG1, D1M1DA

Description: These subroutines use the arithmetic mean of two input values as the independent variable for linear interpolation. They require a doublet or two singlet arrays respectively.

Restrictions: See D1DEG1 or D1D1DA as they are called on respectively.

Calling Sequences:

```
M       CALL D1MDG1 (X1,X2,A(IC),Y)
M       CALL D1M1DA (X1,X2,AX(IC),AY(IC),Y)
```

Subroutine Names: D1M1WM, D1M1MD

Description: These subroutines use the arithmetic mean of two input values as the independent variable for linear interpolation. The interpolated answer is multiplied by the Z value prior to being returned as Y.

Restrictions: Same as D1MDG1 or D1M1DA and Z must be a floating point number.

Calling Sequence:

```
M       CALL D1M1WM(X1,X2,A(IC),Z,Y)
M       CALL D1M1MD(X1,X2,AX(IC),AY(IC),Z,Y)
```

Subroutine Names: D1DG1I, D1D1IM, D1D1MI

Description: These subroutines perform single variable linear interpolation on an array of X's to obtain an array of Y's. D1D1IM multiplies all interpolated values by a constant Z value while D1D1MI allows a unique Z value for each X value. They all call on D1DEG1.

Restrictions: The number of input X's must be supplied as the integer N and agree with the number of Y and Z locations where applicable. Z values must be floating point numbers.

Calling Sequence:

```
M       CALL  D1DG1I  (N,X(DV),A(IC),Y(DV))
M       CALL  D1D1IM  (N,X(DV),A(IC),Z,Y(DV))
M       CALL  D1D1MI  (N,X(DV),A(IC),Z(DV),Y(DV))
```

Subroutine Names: D11DAI, D11DIM, D11MDI

Description: These subroutines are virtually identical to D1DG1I, D1D1IM and D1D1MI respectively. The difference is that they require singlet arrays for interpolation and call on D1D1DA.

Restrictions: Same as D1DG1I, D1D1IM and D1D1MI.

Calling Sequence:

```
M       CALL D11DAI  (N,X(DV),AX(IC),AY(IC),Y(DV))
M       CALL D11DIM  (N,X(DV),AX(IC),AY(IC),Z,Y(DV))
M       CALL D11MDI  (N,X(DV),AX(IC),AY(IC),Z(DV),Y(DV))
```

Subroutine Name: D2DEG1

Description: This subroutine performs two linear interpolations using a bivariate array.

Restrictions and Guidance: The bivariate array must be entered in an ARRAY DATA block using the bivariate array format, (Section 3.7.2). If an independent variable value outside the dependent variable range in the array is supplied, the appropriate end-point value is returned. Independent-variable values supplied may be literal floating point numbers, program variables, user constants or elements in user arrays.

Calling Sequence:

```
M       CALL D2DEG1 (X,Y,BA(IC),Z)
```

where:

X ..... Independent variable value
Y ..... Independent variable value
BA(IC) ... Bivariate array reference of the form smn.An or An
Z ..... Dependent variable value returned

Subroutine Name: D2MXD1

Description: This subroutine is identical to D2DEG1 except that it uses the arithmetic mean of two X values as the independent variable.

Restrictions: Same as D2DEG1

Calling Sequence:

```
M       CALL D2MXD1(X1,X2,Y,BA(IC),Z)
```

Subroutine Name: D2MX1M

Description: This subroutine is identical to D2MXD1 except that the Z value is multiplied by W before being returned.

Restrictions: Same as D2MXD1, W must be a floating point value.

Calling Sequence:

```
M       CALL D2MX1M(X1,X2,Y,BA(IC),W,Z)
```

Subroutine Name: D1IMD1, D1IMWM, D1IMIM

Description: These are indexed subroutines which use the arithmetic mean of two input values as the independent variable for linear interpolation. The array of answers (Y) produced are left as is (D1IMD1), are all multiplied by a single factor (D1IMWM), or each answer is multiplied by a separate factor.

Restrictions: The interpolation array addressed must have an even number of input values and the independent variables must be in ascending order. These routines call up D1D1WM. N is the number of times the operation is to be performed.

Calling Sequence:

```
M       CALL D1IMD1(N,X1(DV),X2(DV),A(IC),Y(DV))
M       CALL D1IMWM(N,X1(DV),X2(DV),A(IC),Z,Y(DV))
M       CALL D1MIM(N,X1(DV),X2(DV),A(IC),Z(DV),Y(DV))
```

6.3.4 Interpolation Using Cyclical Arrays

Subroutine Name: DA11MC, CSIFLX

Description: These subroutines perform linear interpolations using cyclical arrays and multiply the result by a Z-value. CSIFLX is like DA11MC except that for each independent variable there are two dependent variables. For both routines. scale factors can be input for each of the dependent variables.

Restrictions and Guidance: Both routines may be called from any logic block. DA11MC requires two singlet arrays in the ARRAY DATA block. CSIFLX requires a singlet array for the independent variable and a doublet array for the dependent variable. The independent variable array range must be equal to or greater than the value of the cyclic period supplied, and values must be input in ascending order. Any X-values in the array that are greater than the period will be ignored. Independent variable and Z-values supplied may be literal floating point numbers, program variables, user constants, or elements of user arrays.

Calling Sequences

```
M      CALL DA11MC (PER,X,AX(IC),AY(IC),Z,Y)
```

where:

PER ..... period of cyclic function
X ..... Independent variable value supplied
AX(IC) ... Independent variable array reference of the form smn.An or An
AY(IC) ... Dependent variable array reference of the form smn.An or An
Z ..... Value that result is multiplied by before being returned as Y
Y ..... Dependent variable result

Calling Sequences

```
M      CALL CSIFLX(PER,X,AX(IC),AY(IC,IB),Z,YS,YI,Y)
```

where:

PER ..... period of cyclic function
X ..... Independent variable value supplied
AX(IC) ... Independent variable array reference of the form smn.An or An
AY(IC,IB) Dependent variable array reference of the form smn.An or An structured as follows: AY(1,1),AY(1,2),AY(2,1),AY(2,2) ...
Z ..... Scaling factor for the sum of the scaled dependent variables
YS ..... Scale factor for the first dependent variable AY(IC,1)
YI ..... Scale actor for the second dependent variable AY(IC,2)
Y ..... Dependent variable result

Subroutine Name: D11MCY

Description: This subroutine is virtually identical to D11CYL except that the interpolation is multiplied by the floating point Z value prior to being returned as Y.

Restrictions: Calls subroutine D1DEG1.

Calling Sequence:

```
M       CALL D11MCY(PR,X,A(IC),Z,Y)
```

Subroutine Names: D11CYL, DA11CY

Description: These subroutines reduce core storage requirements for cyclical interpolation arrays. The arrays need cover one period only, and the period (PR) must be specified as the first argument. Linear interpolation is performed, and the independent variables must be in ascending order.

Restrictions: All values must be floating point. Subroutines INTRFC is called on by both D11CYL and DA11CY, then D1DEG1 or D1D1DA respectively.

Calling Sequence:

```
M       CALL D11CYL(PR,X,A(IC),Y)
M       CALL DA11CY(PR,X,AX(IC),AY(IC),Y)
```

Subroutine Names: D12CYL, DA12CY

Description: These subroutines are virtually identical to D11CYL and DA11CY except that parabolic interpolation is performed.

Restrictions: See D11CYL and DA11CY. Subroutines LAGRAN and LGRNDA respectively are called.

Calling Sequence:

```
M       CALL D12CYL(PR,X,A(IC),Y)
M       CALL DA12CY(PR,X,AX(IC),AY(IC),Y)
```

Subroutine Names: D12MCY, DA12MC

Description: These subroutines are virtually identical to D11MCY and DA11MC except that parabolic interpolation is performed.

Restrictions: Calls on subroutines LAGRAN and LGRNDA respectively.

Calling Sequence:

```
M       CALL D12MCY(PR,X,A(IC),Z,Y)
M       CALL DA12MC(PR,X,AX(IC),AY(IC),Z,Y)
```

6.3.5 Interpolation Using TARRAY data

Subroutine Names: DT11MC, D11MDT

Description: Subroutines DT11MC and D11MDT do cyclical and linear interpolation respectively of TARRAY data. DT11MC works in the same manner as DA11MC. D11MDT works in the same manner as D11MDA. The difference is the manner in which the array information is passed. Since TARRAYs are stored in a different manner than arrays input in ARRAY DATA, routines which use TARRAY data can not use an array reference such as A10. These routines must be passed just the actual number of the array (eg. 10). The interpolation routines then internally convert this number to a relative number that makes sense in the TARRAY environment.

Restrictions and Guidance: All array references must be the actual array numbers appearing in TARRAY DATA. Beyond that the restrictions of D11MDA and DA11MC apply.

Calling Sequences:

```
M     CALL D11MDT('SMN',X,IC,IC,Z,Y)
M     CALL DT11MC('SMN',PER,X,IC,IC,Z,Y)
```

where:

SMN ..... submodel name
PER ..... period
X ..... Value of independent variable
IC ..... actual array number
Z ..... Value result is multiplied by before being returned as Y
Y ..... Dependant variable result

6.3.6 Generalized Parabolic Interpolation

Subroutine Names: D1DEG2, D1D2DA

Description: These subroutines perform single variable parabolic interpolation. The first requires a double array of X,Y pairs while the second requires singlet arrays of X and Y values. They call on subroutines LAGRAN and LGRNDA respectively.

Restrictions: See LAGRAN or LGRNDA respectively.

Calling Sequence:

```
M     CALL D1DEG2(X,A(IC),Y)
M     CALL D1D2DA(X,AX(IC),AY(IC),Y)
```

Subroutine Names: D1D2WM, D12MDA

Description: These subroutines perform single variable parabolic interpolation by calling on LAGRAN or LGRNDA respectively. However, the interpolated answer is multiplied by the valued addressed as Z prior to being returned as Y.

Restrictions: Same as LAGRAN or LGRNDA and Z must be a floating point number.

Calling Sequence:

```
M     CALL D1D2WM(X,A(IC),Z,Y)
M     CALL D12MDA(X,AX(IC),AY(IC),Z,Y)
```

Subroutine Names: D1MDG2, D1M2DA

Description: These subroutines use the arithmetic mean of two input values as the independent variable for parabolic interpolation. They require a double or two singlet arrays respectively.

Restrictions: See LAGRAN or LGRNDA as they are called on respectively.

Calling Sequence:

```
M     CALL D1MDG2(X1,X2,A(IC),Y)
M     CALL D1M2DA(X1,X2,AX(IC),AY(IC),Y)
```

Subroutine Names: D1M2WM, D1M2MD

Description: These subroutines use the arithmetic mean of two input values as the independent variable for parabolic interpolation. The interpolated answer is multiplied by the Z value prior to being returned as Y.

Restrictions: Same as D1MDG2 or D1M2DA and must be a floating point number.

Calling Sequence:

```
M     CALL D1M2WM(X1,X2,A(IC),Z,Y)
M     CALL D1M2MD(X1,X2,AX(IC),AY(IC),Z,Y)
```

Subroutine Name: D2DEG2

Description: This subroutine performs parabolic interpolation on bivariate arrays. The arrays must be formatted as shown in Section 3.7.2, Bivariate array format.

Restrictions: N.GT.3, M.GT.3 See Bivariate array format.

Calling Sequence:

```
M     CALL D2DEG2(X,Y,BA(IC),Z)
```

Subroutine Name: D2D2WM

Description: This subroutine performs parabolic interpolation on a bivariate array by calling on D2DEG2. The result of the interpolation is multiplied by the W value prior to being returned as Z.

Restrictions: Same as D2DEG2. W must be a floating point number.

Calling Sequence:

```
M     CALL D2D2WM(X,Y,BA(IC),W,Z)
```

Subroutine Name: D2MX2M

Description: This subroutine is virtually identical to D2D2WM except it uses the arithmetic mean of two X values as the independent variable for interpolation.

Restrictions: Same as D2D2WM.

Calling Sequence:

```
M     CALL D2MX2M(X1,X2,Y,BA(IC),W,Z)
```

Subroutine Name: D2MXD2

Description: This subroutine is virtually identical to D2DEG2 except that the arithmetic mean of two X values is used as the independent variable for interpolation.

Restrictions: Same as D2DEG2.

Calling Sequence:

```
M     CALL D2MXD2(X1,X2,Y,BA(IC),Z)
```

6.3.7 Generalized Lagrangian Interpolation

Subroutine Names: LAGRAN, LGRNDA

Description: These subroutines perform interpolation of up to order 50. The first requires one doublet array of X, Y pairs while the second requires two singlet arrays, one of X's and the other of Y's. They contain an extrapolation feature such that if the X value falls outside the range of the independent variable the nearest dependent Y variable value is returned and no error is noted.

Restrictions: All values must be floating point except N which is the order of interpolation plus one and must be an integer. The independent variable values must be in ascending order.

Calling Sequence:

```
M     CALL LAGRAN(X,Y,A(IC,N)
M     CALL LGRNDA(X,Y,AX(IC),AY(IC),Y)
```

Note: A doublet array is formed as follows:

```
X1,Y2,X2,Y2,X3,Y3.....,XN,YN
```

and singlet arrays are formed as follows:

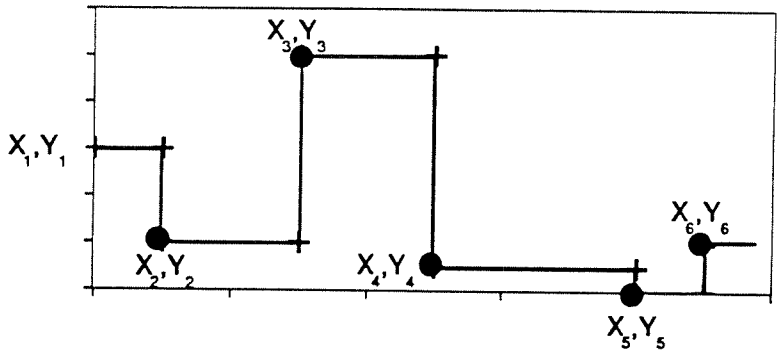
```
X1,X2,X3,....,XN
Y1,Y2,Y3,....,YN
```

6.3.8 Step Interpolation

Subroutine Name: STP1AS

Description: This subroutine performs step interpolation on a doublet array of X,Y pairs. Step interpolation minimizes the input data and interpolation calculation time.

Consider the following example:



Only the points shown in the above example need be input. For independent variable values less than X1, Y1 is returned as the answer. For independent variable values greater than X6, Y6 is returned as the answer. Otherwise for Xi .LE. X .LT. Xi+1 the value Yi is returned as the answer.

Restrictions: All X values must be floating point numbers. The X independent variable values in the doublet array A must be in ascending order.

Calling Sequence:

```
M     CALL STP1AS (X,A(IC),Y)
```

where:

X ..... Input value of independent variable
A ..... Doublet array of X,Y pairs
Y ..... Output value of dependent variables

Subroutine Name: STP2AS

Description: This subroutine performs step interpolation on a pair of singlet arrays containing corresponding values of X and Y.

Restrictions: All X values must be floating point numbers. The X independent variable values in the AX array must be in ascending order. The number of values in the AX and AY arrays must be the same.

Calling Sequence:

```
M     CALL STP2AS(X,AX(IC),AY(IC),Y)
```

where:

X ..... Input value of the independent variable
AX ..... Singlet array of X values
AY ..... Singlet array of Y values corresponding to the X values in AX
Y ..... Output value of the dependent variable

Subroutine Name: STP1AM

Description: This subroutine performs step interpolation on a doublet array A of X,Y pairs using a singlet array of X's (independent variable input values) to form a singlet array of Y's. The X independent variable values in the double array A must be in ascending order.

Restrictions: The number of input X's must be supplied as the integer N and agree with the number of Y locations. X values must be floating point numbers.

Calling Sequence:

```
M     CALL STP1AM(N,X(DV),A(IC),Y(DV))
```

Subroutine Name: STP2AM

Description: This subroutine is virtually identical to STP1AM except the interpolation is performed on a pair of singlet arrays of AX and AY.

Restrictions: Same as STP1AM.

Calling Sequence:

```
M     CALL STP2AM(N,X(DV),AX(IC),AY(IC),Y(DV)
```

6.4 Output Subroutines

6.4.1 Printout of Nodal Attributes

Subroutine Names: CPRINT, QPRINT, TPRINT, GPRINT, TDUMP

Description: These subroutines write values of thermal capacitance, conductance Q-source and temperatures to the system print file. All temperatures and non-zero conductance, capacitance and Q-source values are printed for a specified submodel. TDUMP is just like TPRINT except that it does not print out the balance criteria header or page between submodels. It just prints the submodel name and the time for each header.

Restrictions and Guidance: All routines are callable from OUTPUT CALLS, but not restricted to this. A submodel name must be specified and should agree with the submodel name on the preceding HEADER OUTPUT CALLS record.

Calling Sequences:

```
CALL CPRINT ('SMN')
CALL QPRINT ('SMN')
CALL TPRINT ('SMN')
CALL GPRINT ('SMN')
CALL TDUMP  ('SMN')
```

where SMN is either a submodel name or 'ALL' to print all submodels. Character delineators (') are required.

6.4.2 Printout of Network Connectivity/Heat Flow Map

Subroutine Name: QMAP

Description: This subroutine computes and writes out a *thermal* network connectivity and heat flow map based upon current nodal temperatures and conductance values. If called prior to a solution routine (e.g., STDSTL) a node-conductor connectivity map will be output that gives the values of all conductors connected to each node expressed both as an absolute conductance value and as a percentage of the total connected to the node. If called after a solution routine, the heat flow through each conductor will be output also. QMAP may be used to map a single submodel or all submodels. If the QMAP file name is set in OPTIONS DATA, the map will be written to this file. Output can optionally be routed to the OUTPUT file, although this copy can be suppressed if a QMAP file is named.

Restrictions and Guidance: Because of the volume of output, QMAP calls should all appear in OPERATIONS DATA and must never appear in a VARIABLES or FLOGIC block. If called prior to a solution routine, a QMAP call should be preceded by calls to subroutines GVTEMP and GVTIME in order to initialize the conductor values.

Calling Sequence:

```
CALL QMAP ('SMN', 'ARGS', NPRNT)
```

where:

SMN	a thermal submodel name
ARGS	character string containing commands (see below)
NPRNT	OUTPUT print flag: 0 = print to OUTPUT file, 1 = suppress printing to OUTPUT file (but still write to QMAP file if named)

where 'ARGS' can be one of the following values:

'QDAHB'	Heat flow map plus conductance map for all node types
'ALL'	same as above
'Q'	Heat flow map only - all node types
'D'	Heat flow and conductance map - diffusion nodes only
'A'	Heat flow and conductance map - arithmetic nodes only
'B'	Heat flow and conductance map - boundary nodes only
'H'	Heat flow and conductance map - heater nodes only
'XXXX'	XXXX = some combination of Q,D,A,H and/or B

Note: character arguments must be enclosed by character delineators(').

Subroutine Name: NODMAP

Description: This subroutine is the same as QMAP, but only produces a mapping of one node. This routine can be used in place of QMAP to reduce the amount of outputs. Both heat flow and conductance maps are printed.

Calling Sequence:

```
CALL NODMAP ('SMN', NODE, NPRNT)
```

where:

SMN ..... a thermal submodel name
NODE ..... ID of node to be mapped
NPRNT .... OUTPUT print flag: 0 = print to OUTPUT file, 1 = suppress printing to OUTPUT file (but still write to QMAP file if named)

6.4.3 Restart Data Save Subroutine

Subroutine Name: RESAVE

Description: (See Section 4.3) This subroutine writes all program variables (temperatures, capacitances, conductances, Q-sources, control constants, user constants, user arrays, and flow data) to the binary restart (RSO) file. RESAVE requires that a user's permanent file name be equated to the name RSO in the OPTIONS DATA. Each time RESAVE is called, a sequential restart record number and current problem time will be printed for future reference. At the user's option, parts of the program variables (e.g., capacitances, Q-sources) may be left out of the output process.

Restrictions and Guidance: Should never be called from a VARIABLES or FLOGIC block. Since its output is not restricted to a single submodel, calls to RESAVE should appear in only one OUTPUT CALLS block in a multi-submodel problem. In this case, the user controls the frequency of output to RSO with the control constants OUTPUT and/or ITEROT for the submodel associated with the OUTPUT CALLS block that contains the call to RESAVE. The "ALL" option should be used unless the RSO file threatens to grow to an unmanageable size. To simplify program control, *named (global) user constants are never saved.*

Calling Sequence:

```
CALL RESAVE ('ARGS')
```

where 'ARGS' is a combination of characters that define the types of data to be written to the restart file. It works as follows:

'ALL' ..... All parameters are written to RSO
'T' ..... Write Temperatures to RSO
'C' ..... Write Capacitance values to RSO
'Q' ..... Write Q-source values to RSO
'G' ..... Write Conductor values to RSO
'N' ..... Write control constants to RSO
'U' ..... Write *numbered* user constants to RSO
'A' ..... Write user arrays and CARRAYs to RSO
'L' ..... Write lump PL, TL, XL, and QDOT to RSO (for plots)
'P' ..... Write path FR to RSO (for plots)
'F' ..... Write remaining FLUINT data to RSO (for restarts)
'xxxxxx' ..... xxxxxx is any combination of T,C,Q,G,U,N,A,L,P, and/or F

Note: ARGS must always be enclosed in character delineators ('). In order to restart FLUINT models, ARGS must be 'ALL' or include 'LPF'.

Subroutine Name: CRASH

A variation of RESAVE is available for creating and updating crash files (i.e., files that assist in the recovery from an unexpected run termination). The routine, CRASH, is equivalent to calling RESAVE with an argument of 'ALL', *except that repeated calls overwrite the previously stored snapshot.* Therefore, CRASH can be called an arbitrary number of times without causing the crash file size to grow. Also, the contents (file buffers) are flushed each call, so that a subsequent failure or abort does not render the last saved state invalid.

The cost of a call to CRASH is not completely insignificant; the user must trade–off how often to call it versus how much ground must be retraced in the event of a crash. Calling it from FLOGIC 0 or VARIABLES 0 will maximize both its cost and its "freshness," while calling it from OPERATIONS DATA will minimize both. OUTPUT CALLS is probably the most logical compromise location for most uses. Often, not all OUTPUT CALLS blocks are used for all submodels; using such a otherwise unused block enables the user to customize the frequency of calls to CRASH.

Description: Saves one snapshot of the entire model for later use in restarting. Subsequent calls overwrite the previously saved snapshot, conserving file size. CRASH is equivalent to a call to RESAVE with an argument 'ALL', except only the most recent snapshot is saved if repeated calls are made. CRASH may be called from any logic block.

The crash file is closed after each call, and is reopened with STATUS='UNKNOWN.' The first call to CRASH will generate an output file message containing the record number: "CRASH FILE RECORD NUMBER NNN." This record number should be used in subsequent calls to RESTAR, with the crash file name input as the RSI file in OPTIONS DATA.

Calling Sequences:

```
CALL CRASH(cfname)
```

where:

`cfname` .... crash file name (including path if desired) in single quotes, or CARRAY reference.

On case–sensitive UNIX machines, use the CARRAY option to preserve lower case letters in file names. Otherwise, the file name will be capitalized.

Examples:

```
CALL CRASH('OHNO')        $ LOCAL FILE NAME
CALL CRASH(UCA10)         $ CARRAY #10 CONTAINS FILE NAME
```

6.4.4 Variable Save Routine

Subroutine Name: SAVE

Description: (See Section 4.3) This subroutine writes all program variables (temperatures, capacitances, conductances, Q-sources, control constants, user constants, user arrays, and flow data) to the SAVE file. SAVE requires that a user's permanent file name be equated to the name SAVE in the OPTIONS DATA. At the user's option, parts of the program variables (e.g., capacitances, Q-sources) may be left out of the output process. At the user's option the control constants and energy balance criteria will be printed when the subroutine is called.

Restrictions and Guidance: Should never be called from a VARIABLES or FLOGIC block. Since its output is not restricted to a single submodel, calls to SAVE should appear in only one OUTPUT CALLS block in a multi-submodel problem. In this case, the user controls the frequency of output to SAVE with the control constants OUTPUT and/or ITEROT for the submodel associated with the OUTPUT CALLS block that contains the call to SAVE. The 'ALL' option should be used unless the SAVE file threatens to grow to an unmanageable size. To simplify program control, *named (global) user constants are never saved.*

Calling Sequence:

```
CALL SAVE ('ARGS',NFLAG)
```

where 'ARGS' is a combination of characters that define the types of data to be written to the restart file. It works as follows:

'ALL' ..... All parameters are written to SAVE
'T' ..... Write Temperatures to SAVE
'C' ..... Write Capacitance values to SAVE
'Q' ..... Write Q-source values to SAVE
'G' ..... Write Conductor values to SAVE
'N' ..... Write control constants SAVE
'U' ..... Write *numbered* user constants to SAVE
'A' ..... Write user arrays and CARRAYs to SAVE
'L' ..... Write lump PL, TL, XL, and QDOT to SAVE (for plots)
'P' ..... Write path FR to SAVE (for plots)
'F' ..... Write remaining FLUINT data to SAVE (for restarts)
'xxxxxx' ..... xxxxxx is any combination of T,C,Q,G,U,N,A,L,P, and/or F

Note: ARGS must always be enclosed in character delineators ('). In order to restart FLUINT models, ARGS must be 'ALL' or include 'LPF'. NFLAG controls whether or not the current values of the control constants are printed when TSAVE is called. If NFLAG = 1, the control constants and model energy balance data will be printed. If NFLAG = 0, only the message "SAVE called at TIMEN = xxxxx" will appear.

6.4.5 Saving Data for Parametric Cases

Subroutine Name: SAVPAR

Description: (See Section 4.3) This subroutine saves all program variables (temperatures, capacitances, conductances, Q-sources, control constants, user constants, user arrays, and all flow data) to an internal program file PARAM. Each call to SAVPAR returns a record number. This is in record number where SAVPAR began its write. Each call to SAVPAR returns a different record number. Subroutines SAVPAR and RESPAR replace the more restrictive Initial Parameters and Final Parameters features used on prior versions of SINDA.

Restrictions and Guidance: Calls to SAVPAR are probably most useful in the OPERATIONS DATA block. Calls from OUTPUT CALLS may also be made. Calls from a VARIABLES or FLOGIC block are not appropriate. If a solution routine (e.g., FWDBCK) call was preceded and followed by calls to SAVPAR, this would make the "initial parameters" (before FWDBCK) and the "final parameters" (after FWDBCK) available for subsequent OPERATIONS DATA calls to RESPAR. To simplify program control, *named (global) user constants are never saved.*

Calling Sequence:

```
CALL SAVPAR (NREC)
```

NREC is an integer variable that identifies the data package that is written to PARAM. NREC will be used by subroutine RESPAR to restore the data. Each call to SAVPAR will return a unique argument. This argument should be a user constant (e.g., K4, ATEST) or an array reference.

6.4.6 Temperature Save Subroutine

Subroutine Name: TSAVE

Description: This subroutine saves nodal temperatures and problem time on a permanent file for use in post processors such as temperature history plot programs. At the users option, the control constants and energy balance criteria will be printed or not printed when the subroutine is called. TSAVE is a specialized call to the SAVE subroutine. TSAVE requires that a user's permanent file name be equated to the name SAVE in OPTIONS DATA.

Restrictions and Guidance: A call to TSAVE writes the temperatures for the entire model (all thermal submodels) to the SAVE file. Each set of temperatures is preceded by the number of active nodes in the model and the problem time, TIMEN. Calls to TSAVE are inappropriate in any VARIABLES block. Since all the temperatures are written, TSAVE should be called from only one OUTPUT CALLS block in a multi-submodel problem. The values of control constants OUTPUT and/or ITEROT for this submodel will control the frequency of TSAVE outputs.

Calling Sequence:

```
CALL TSAVE (NFLAG)
```

NFLAG controls whether or not the current values of the control constants are printed when TSAVE is called. If NFLAG = 1, the control constants and model energy balance data will be printed. If NFLAG = 0, only the message "SAVE called at TIMEN = xxxxx" will appear.

6.4.7 Printing Minimum and Maximum Temperatures

Subroutine Name: TMNMX

Description: This subroutine prints out the maximum and minimum temperatures of either all the nodes in a submodel or of a specified list of nodes in a thermal submodel. The user may also specify one or more time intervals for TMNMX to be in effect. The minimum and maximum temperatures are printed at the end of the time interval(s) specified.

Restrictions and Guidance: Array ATM is an array which contains the floating point times at which the maximums and minimums are to be output. For example, if the argument were (A1) and array 1 contained the following entries: 1.0, 3.0, 3.5, END, then three outputs would occur, the first at TIMEN = 1.0 covering the interval TIMEO to 1.0; the second at 3.0 for the interval 1.0 to 3.0; and the final output at 3.5 for the period 3.0 to 3.5. For a printout at every time stop, enter zero (0) for ATM.

The next three arguments describe the node numbers for which the maximum and minimum temperatures are desired. These node numbers are input via a CARRAY DATA block. SMN tells which submodel's CARRAY DATA block contains the node numbers. NCA1 is the actual number of the first CARRAY to use. NUMCAS is the number of carrays to use. This is the same argument arrangement used in BNDGET and BNDDRV. The descriptions of those routines contain examples of how this is done.

Calling Sequence:

```
M       CALL TMNMX (ATM(IC),"SMN",NCA1,NUMCAS)
```

where:

ATM ..... an array of output times or 0 for all output times
SMN ..... submodel name containing CARRAY NCA1
NCA1 ..... actual number of first CARRAY to use
NUMCAS ... number of sequential CARRAYs to use

6.4.8 Array Data Printout

Subroutine Name: PRINTA

Description: This subroutine allows the user to print out an array of values, five to the line. The integer array length N and the first data value location must be specified. Each value receives an indexed label. The user must supply a six-character alphanumeric word L to be used as a common label and an integer value M to begin the index count.

Restrictions: The array values to be printed must be floating point numbers. If L is supplied as a literal Hollerith data value (instead of a reference to a user constant containing same), it must be entered in character type notation (e.g., 'TEMP').

Calling Sequence:

```
      CALL PRINTA (L,A(DV),N,M)
```

If the label L were the word 'TEMP', N were 3 and M were 6, the line of output would look as follows:

```
TEMP    (  6)  value TEMP      (  7)  value TEMP        (  8)  value
```

Subroutine Name: GENOUT

Description: This subroutine prints out whole arrays or parts of arrays, printing up to 10 values per line. The numbers may be real, integer, or both.

Calling Sequence:

```
M     CALL GENOUT(A(IC),ISTRT,ISTP,'NAME')
```

where:

A	array containing numbers to be printed
ISTRT	location of the first value to be printed (ISTRT .GE. 1).
ISTP	location of the last value to be printed (ISTP .GT. ISTRT and .LE. the integer count of A plus one)
'NAME' ...	title of up to 22 characters for identification

Guidance: Recall that the first cell in every SINDA array is equivalenced to its integer count. To avoid printing this first cell, use ISTRT=2. Similarly, to include the last value of the array, use ISTP = one plus the integer count of the array (i.e., 1+NA(IC)). The first character of NAME may be used for carriage control. That is, if the character is a blank, printout appears immediately, within other output. Any non-blank character will begin the printout at the top of the new page.

6.4.9 Numerical Differencing Characteristics Printout

Subroutine Name: STNDRD

Description: Subroutine STNDRD causes a line of output to be printed giving the present time, the last time step used, the most recent CSGMIN value, the maximum diffusion temperature change calculated over the last time step and the maximum relaxation change calculated over the last iteration. ANN refers to the actual node number that caused the appropriate value. The line of output looks as follows:

```
*    *    *    *
TIME      DTIMEU      CSGMIN(ANN)      TEMPCC(ANN)      RELXCC(ANN)
```

Restriction: Same as TPRINT.

Calling Sequence:

```
CALL STNDRD ('SMN')
```

where SMN is a submodel Name

6.4.10 Sorted Temperature Print

Subroutine Name: SORTPR

Description: This subroutine prints out nodal temperatures. These temperatures may appear sorted by node number or by temperature.

Restrictions and Guidance: This routine is usually called from OUTPUT CALLS data, but is not restricted to this. A submodel name must be specified. To print out all submodels the word 'ALL' may appear in place of a submodel name. The print flag IFLAG must be 0 or 1. If IFLAG=0 temperatures will be sorted by node number. If IFLAG=1 output will be sorted by temperature.

Calling Sequence:

```
CALL SORTPR ('SMN', IFLAG)
```

where:

SMN ..... a thermal submodel name
IFLAG .... print flag: 0 = print by node, 1 = print by temperature

6.4.11 Save Temperatures for Boundary Temperature Driving

Subroutine Name: BNDGET

Description: Subroutine BNDGET gets temperatures from one or more SAVE files and writes them to an ASCII output file. The temperatures to fetch are specified by a list of submodel names and node numbers appearing in CARRAY DATA. The call to BNDGET also specifies the start and end times. In this way data for a specific segment of time may be extracted from the SAVE file. Data from any number of SAVE files may be extracted by using multiple calls to BNDGET. Multiple calls may also be used to extract data for a number of different time segments on the same SAVE file. The data may be transferred to a number of independent output files if a different FILOUT name is used on each call. A number of sets of data may be sent to a single output file if FILOUT is the same for a series of calls, because FILOUT is not rewound prior to the write commands. In this case, it is the users responsibility to be sure that there are no overlaps among the time segments written out.

Restriction: BNDGET should be called in OPERATIONS DATA. It should be called after all relevant solution routines have been performed if it is to use a currently active SAVE file. It can also be used during initialization to prepare a file for BNDDRV from existing SAVE files.

The user should be aware that if BNDGET is called more than once in a single run, a new 'FILOUT' name must be used for each call. Otherwise, multiple files will be written to the output file and BNDDRV will fail to work. BNDGET can use any set of CARRAY data. The only restriction is that the arrays to be used must appear sequentially in the names block.

Calling Sequence:

```
M       CALL BNDGET (NUMCAS, SMN, NCA1, FILIN, FILOUT,
     .          TIME1, TIME2, TIMDEL, FIGNAM, PFLAG)
```

where:

NUMCAS ... Number of CARRAYs to use (integer)
SMN ..... Submodel name where CARRAY data appears (character)
NCA1 ..... Actual number of 1st CARRAY to use (integer)
FILIN ..... A save file name for input (character)
FILOUT ..... name of output file (character)
TIME1 ..... Start time of read from FILIN (real)
TIME2 ..... End time of read (real)
TIMDEL .... A delta to add to read times (real) FIGNAM configuration name from relevant BUILD record or the word 'ALL' (character)
PFLAG ..... Print flag if not 0 then writes out TIME1, TIME2, FILIN, TIMDEL (Integer)

Example: The following shows the relevant data blocks and their use with BNDGET.

```
HEADER OPTIONS DATA
          SAVE = NAME.TSV
HEADER CARRAY DATA, MAIN
          1 = TEST
          2 = MOD1, 100, 200, 300, MOD2, 100, 200, 300
          3 = 400, 500, MOD3, ALL
C    The word ALL may appear to indicate all nodes in
C    a submodel
          4 = TEST
HEADER OPERATIONS DATA
BUILD MODELS, MOD1, MOD2, MOD3
                .
                .
                .
      CALL BNDGET(2,'MAIN',2,'NAME.TSV','NAME.BND',
     .          0.0,1.0,0.0,'MODELS',0)
```

6.4.12 Print Heater Node Q Values

Subroutine Name: HNQPNT

Description: This subroutine internally calls HNQCAL. It then prints out the Q values calculated for the heater nodes.

Restrictions and Guidance: This routine is callable from OUTPUT CALLS but is not restricted to this. A thermal submodel name must be specified and should agree with a submodel name appearing on the a BUILD card.

Calling Sequence:

```
CALL HNQPNT ('SMN')
```

where SMN is a thermal submodel name, or use the word 'ALL' to print all thermal submodels. Character delineators (') are required.

6.4.13 Print Routine Controller

Subroutine Name: PRINT

Description: Given the name of a *thermal* print routine to control, this routine will control the number of times the print routine is called. In addition, if a temperature print routine is being controlled the name of a temperature conversion routine may also be passed in. The passed in routine will perform the required conversions for output.

Restrictions and Guidance: This routine is callable from OUTPUT CALLS but is not restricted to this. A thermal submodel name must be specified and should agree with a submodel name appearing on the BUILD card. The submodel name is the only character argument. All other arguments are either subroutine references or integers.

Calling Sequence:

```
CALL PRINT ('SMN', PFUNC, NSKIPS, CONV)
```

where:

SMN	a thermal submodel name or the word 'ALL'. Character delineators (') are required.
PFUNC	The name of the print routine to control. Valid entries are: TPRINT,QPRINT,CPRINT,GPRINT. This argument must *not* be in quotes.
NSKIPS ...	Number of times PRINT should be called with SMN before PFUNC is called.
CONV	The name of a conversion routine. Valid entries are: CTF, CTK, FTC, FTK, FTR, KTC, KTF, and RTF. Enter a 0 for no conversion. This argument must *not* be in quotes.

6.4.14 Reset Page Header Information

Subroutine Name: PAGHED

Description: This subroutine gives the user access to the number of lines on a page and the titles on the page. The user may also use this routine to add extra lines to the standard output file NOUT without disrupting the starting point of new pages.

MLINE is the maximum number of output lines per page, and is normally set once in OPTIONS DATA. LINDEL allows the user to add lines to the output file NOUT and to increment the line counter accordingly to avoid disrupting the starting point for new pages. TITLE, the main page title, is similarly normally set once in OPTIONS DATA. STITLE is normally set by solution routines and mapping routines.

Calling Sequence:

```
CALL PAGHED(MLINE, LINDEL, TITLE, STITLE)
```

where:

MLINE .... the number of lines per page (an integer). If MLINE is 0 then the value of MLINE will not be changed.

LINDEL ... a number to increase the current line counter. If the current line counter plus LINDEL is greater or equal to MLINE then a new page will be started, using current MLINE, TITLE, and STITLE values as supplied by the current call to PAGHED.

TITLE .... the main page title (single quotes are required, or TITLE can be a CARRAY entry such as UCA10). If this argument is a space, (' '), the current TITLE will not be changed.

STITLE ... the page subtitle (single quotes are required, or STITLE can be a CARRAY entry such as UCA10). The subtitle is set to the name of the current solution routine or mapping routine at the start of those routine, so any calls to PAGHED to customize the subtitle will be overwritten at the start of those routines. If this argument is a space, (' '), the current STITLE will not be changed.

6.4.15 Creating Additional User Files

Subroutine Name: USRFIL

Description: This subroutine opens a named file and returns a valid unit number. The returned unit number can be used in subsequent WRITE or READ statements, and hence should appear in HEADER USER DATA, GLOBAL. This routine is intended to extend the USER1 and USER2 options.

Normally, the program will assign a valid unit number and proceed to open the file as a FORMATTED (ASCII) file. If users wish to open an UNFORMATTED (binary) file or otherwise wishes to perform the OPEN statement themselves, passing the file name as 'UNITONLY' will cause the routine to return a valid unit number without opening the file.

Restrictions: Do not attempt to open the same file twice. The processor will abort if the file could not be opened. Users are cautioned that many operating systems impose a quota on the maximum number of files that may be opened at one time; exceeding this limit will cause the system to abort the processor.

Calling Sequence:

```
CALL USRFIL (NUNIT, FNAME, STATUS)
```

where:

NUNIT .... the returned unit number (an integer). The variable that will contain the unit number should appear in HEADER USER DATA, GLOBAL

FNAME .... the name of the file to be opened (single quotes are required, or FNAME can be a CARRAY entry such as UCA10).
The names 'UNITONLY' and 'unitonly' have special meaning: they signal the routine to assign a valid unit number to NUNIT without actually opening a file.

STATUS ... the status flag to use in the Fortran OPEN statement, in single quotes:
'OLD' if the file exists and it can be overwritten, or
'NEW' if it shouldn't exist yet, producing an abort if it does exist to protect it from being overwritten, or 'UNKNOWN.'
No OPEN operation will be performed if FNAME is 'UNITONLY'.

6.5 Input Subroutines

6.5.1 Boundary Node Temperature Driving

Subroutine Name: BNDDRV

Description: Subroutine BNDDRV drives boundary temperatures. This routine allows the use of temperature history file (generally a FILOUT file from subroutine BNDGET) to provide boundary node temperature histories during execution. The history file may be externally generated if it meets the following requirements (written to an 8F10.3 format).

```
TIME,(TEMP (I), I=1,N)
```

where:

TIME	time point
TEMP (I) ..	temperature of Ith boundary node at TIME
N	number of boundary nodes to be driven

This routine requires an array of the actual numbers of the boundary nodes which are to be driven. The input file is searched and linear interpolation is performed to find the correct temperature for each boundary node in the specified input at TIMEM.

Restrictions and Guidance: BNDDRV should be called from VARIABLES 0. It should be called from the submodel with the smallest time step as it reads in all specified data at one time. Alternately the user may provide CARRAY data for each model and have a call in each VARIABLES 0 block.

Calling Sequence:

```
CALL BNDDRV (NUMCAS, SMN, NCA1, FILIN)
```

where:

NUMCAS ...	number of CARRAYS to use (integer)
SMN	Submodel name where CARRAY data appears (character)
NCA1	Actual Number of 1st CARRAY to use (integer)
FILIN	A permanent file name for input (character)

Example: The following shows the relevant data blocks and their use with BNDGET:

```
HEADER CARRAY DATA, MAIN
         1 = TEST
         2 = MOD1, 100, 200, 300, MOD2, 100, 200, 300
         3 = 400, 500, MOD3, 100
         4 = TEST
HEADER VARIABLES 0, MAIN
               .
               .
               .
         CALL BNDDRV (2, 'MAIN', 2,'MODELS.BND')
               .
               .
```

6.5.2 Re-Initialization for Parametric Cases

Subroutine Name: RESPAR

Description: (See Section 4.3) This subroutine reads in all program variables (temperature, capacitances, conductances, Q-sources, control constants, user constants, user arrays, and all flow data) from program file PARAM.

Restrictions and Guidance: Parameters read in are accessed by record number, (Reference Subroutine SAVPAR). Called generally from OPERATIONS DATA prior to a solution routine. With subroutine STDSTL, conditional calls from VARIABLES 0 are appropriate. This requires logic making calls conditional on pseudo-time, TIMEN. Each call to SAVPAR will be associated with a record number to be used with subroutine RESPAR.

Calling Sequence:

```
CALL RESPAR (NREC)
```

where NREC is the record number associated with the set of parameters to be read in. NREC is an integer variable name or literal.

6.5.3 Reading In Parameters from the Restart File

Subroutine Name: RESTAR

Description: (See Section 4.3.) This subroutine reads in all program variables (temperatures, capacitances, conductances, Q-sources, control constants, user constants, user arrays and flow variables) from the binary RSI (restart input) file. Each call to RESAVE will print an associated record number to be used with subroutine RESTAR. Equivalently, the first call to CRASH will print the associated record number to be used in the call to RESTAR.

The exact composition of the data read in depends on argument 'ARGS' used with subroutine RESAVE when the restart data was written to the RSO file. If being used to access a crash file, the file contains the information equivalent to a call to RESAVE with the argument 'ALL'.

Restrictions and Guidance: Calls to RESTAR require that file RSI be identified with a user's permanent file name in the OPTIONS DATA block. If any network elements, constants, arrays or array elements have been added or deleted, a collision will be detected and a restart cannot be performed. In this case, an error message will be printed and the program will abort. (See RESTNC below.) *Only use a restart file that was created by the same SINDA/FLUINT version as the one being used to read the file.*

Caution: Calls to RESTAR do not affect the dormant/awake status of thermal submodels, and do not affect the location of ties. In other words, the effects of DRPMOD and PUTTIE calls in previous runs are irrelevant. Also, note that the collision checks cannot detect changes in input order, which should be avoided unless purposefully employed by an advanced user.

Calling Sequence:

```
CALL RESTAR (NREC)
```

where NREC is the restart file record number associated with the data to be read in.

Subroutine Name: RESTNC

Description: (See Section 4.3.) This subroutine is identical to RESTAR, except that collision checks are not performed. This routine is intended for advanced manipulations by users sufficiently versed in the code to assume responsibility for collision checks.

Caution: Calls to RESTNC do not affect the dormant/awake status of thermal submodels, and do not affect the location of ties. In other words, the effects of DRPMOD and PUTTIE calls made in previous runs are irrelevant.

6.6 Utility Subroutines

6.6.1 Finding User Array Locations

Subroutine Name: ARYTRN

Description: This routine returns the absolute location in the A-array of a user array, given the actual number of the array. For numeric arrays, the integer count location is returned. For character arrays, the location of the first character in the array is returned. Note: all character arrays elements are 128 characters long.

Restrictions and guidance: This routine bypasses the need to consult the array data storage dictionary when relative locations are required for F-type (Fortran) statement logic.

Calling Sequence:

```
CALL ARYTRN('SMN',NUM,L)
```

where:

SMN ..... a submodel name enclosed in character delimiters (')
NUM ..... an actual user array number
L ..... the returned relative location of the array

6.6.2 Finding Conductor Value Locations

Subroutine Name: CONTRN

Description: This routine returns the relative location in the G-array of a network conductor, given the actual conductor number.

Restrictions & Guidance: This routine bypasses the need to consult the conductor data storage dictionary when relative locations are required for F-type (Fortran) statement logic.

Calling Sequence:

```
CALL CONTRN ('SMN',NUM,L)
```

where:

SMN ..... a submodel name enclosed in character delimiters (')
NUM ..... an actual user conductor number
L ..... the returned relative location of the conductor

6.6.3 Array Integration

Subroutine Name: CRVINT

Description: This subroutine performs an integration of the doublet array, A, and stores the results in doublet array B. The independent variables of the A array are transferred directly to the B array. The dependent variables of the B array are calculated by:

$$B(2) = 0.0$$

$$B(2*N) = B(2*(N-1)) + 0.5*[A(2*N)+A(2*(N-1)] * [A(2*N-1)-A(2*(N-1)-1)] \qquad N=2, NP$$

where NP = number of pairs of points in the A array (half the integer count)

This subroutine was written primarily for integration of specific heat arrays to obtain enthalpy arrays but could be used for integration of any dependent variable.

Restrictions: Space in B array must be exactly equal to the space in the A array. There must be at least two points in A array (i.e., the integer count must be at least 4).

Calling Sequence:

```
M       CALL CRVINT(A(IC),B(IC))
```

6.6.4 Finding CARRAY Locations

Subroutine Name: CRYTRN

Description: This routine returns the index in the UCA array given the actual number of a CARRAY.

Restrictions and Guidance: This routine bypasses the need to consult the CARRAY data storage dictionary when absolute indexes are needed for F-type (FORTRAN) statement logic.

Calling Sequence:

```
CALL CRYTRN ('SMN',NUM,INDEX)
```

where:

SMN ..... a submodel name enclosed in character delimiters (')
NUM ..... an actual user CARRAY number
INDEX .... the returned pointer into the UCA array

6.6.5 Finding Submodel Name Locations

Subroutine Name: MODTRN

Description: This routine returns the relative location of a thermal submodel name in the array of submodel names (as defined by the BUILD records).

Restrictions and Guidance: This routine bypasses the need to consult the submodel name data storage dictionary when submodel sequence numbers are needed for F-type (Fortran) statement logic.

Calling Sequence:

```
CALL MODTRN('SMN',L)
```

where:

SMN ..... a thermal submodel name enclosed in character delimiters (')
L ..... the returned submodel sequence number

6.6.6 Finding Node Attribute Locations

Subroutine Name: NODTRN

Description: This routine returns the relative location in the T,C and Q arrays of nodal temperatures, capacitor and Q-values, given an actual node number.

Restrictions and Guidance: This routine bypasses the need to consult the node data storage dictionary when relative locations are required for F-type (Fortran) statement logic.

Calling Sequence:

```
CALL NODTRN('SMN', NUM,L)
```

where:

SMN ..... a thermal submodel name enclosed in character delimiters (')
NUM ..... an actual user node number
L ..... the returned relative location of the node in arrays T, C, and Q

6.6.7 Finding User Constant Locations

Subroutine Name: NUMTRN

Description: This routine returns the relative location in the K (and equivalenced XK) array, of a numbered user constant, given an actual constant number.

Restrictions and Guidance: This routine bypasses the need to consult the user constant dictionary when relative locations are required for F-type (Fortran) statement logic.

Calling Sequence:

```
CALL NUMTRN('SMN',NUM,L))
```

where:

SMN ..... a submodel name enclosed in character delimiters (')
NUM ..... an actual user constant number
L ..... the returned relative location of the constant

6.6.8 Area Integration

Subroutine Name: TRPZDA

Description: This subroutine performs area integration by the trapezoidal rule. It may be used whether or not the DX increment is uniform, but a Y value corresponding to each X value is required. The operation performed is as follows:

```
A = 1/2(Xi - Xi-1)*(Yi + Yi-1)     ,    i = 2,N
```

All values must be floating point numbers except the array length N which must an integer.

Calling Sequence:

```
CALL TRPZDA(N,X(DV),Y(DV),A)
```

6.6.9 Least Squares Curve Fitting

Subroutine Name: LSTSQU

Description: This subroutine performs a least-squares curve fit to an arbitrary number of X,Y pairs, yielding a polynomial equation of up to order 10. Rather than using a double precision matrix inversion, this subroutine calls the subroutine SIMEQN to obtain a simultaneous solution.

Restrictions: All values must be floating point numbers except N and M which must be integers. N is the order of the polynomial desired and is one less than the number of coefficients desired. M is the array length of the independent X or dependent Y values.

Calling Sequence:

```
CALL LSTSQU(N,M,X(DV),Y(DV),A(DV))
```

6.6.10 Putting a Submodel In a Boundary State

Subroutine Name: DRPMOD

Description: This subroutine drops a *thermal* submodel out of the current pseudo-compute sequence. Conductors will still be attached to these nodes but their temperatures will no longer be changed. Therefore they will act as boundary nodes until they are replaced by subroutine ADDMOD. DRPMOD cannot be used to drop fluid submodels.

Restrictions and Guidance: This routine may called *once* for each *active* thermal submodel to be dropped. *It should only be within OPERATIONS DATA.* Calls to DRPMOD in any VARIABLES, OUTPUT CALLS, or FLOGIC block are not appropriate. Calls to RESTAR or RESTNC do not affect prior calls to DRPMOD—the dormant/awake status of previous runs is ignored.

Calling Sequence:

```
CALL DRPMOD ('SMN')
```

where SMN is a submodel name (character delineator ' is required)

6.6.11 Reactivating a Boundary State Submodel

Subroutine Name: ADDMOD

Description: This subroutine replaces a *thermal* submodel in the pseudo compute sequence. ADDMOD cannot be used with fluid submodels.

Restrictions and Guidance: This routine can be called *once* for each thermal submodel *that been dropped*, returning it to active status. *It should only be within OPERATIONS DATA.* Calls to ADDMOD in any VARIABLES, OUTPUT CALLS, or FLOGIC block are not appropriate. Calls to RESTAR or RESTNC do not affect prior calls to ADDMOD—the dormant/awake status of previous runs is ignored.

Calling Sequence:

```
CALL ADDMOD ('SMN')
```

where SMN is a submodel name (character delineator ' is required)

6.6.12 Calculating Transient Energy Balance

Subroutine Name: COMBAL

Description: This subroutine calculates the energy balance for transient routines.

Restrictions and Guidance: This routine sets the values of EBALSC and EBALNC for all currently active submodels.

Calling Sequence:

```
CALL COMBAL
```

6.7 Applications Subroutines

6.7.1 Heater Node Energy Requirements

Subroutine Name: HNQCAL

Description: This routine calculates the heat necessary to maintain the heater nodes at their set temperatures. The results can be found in the heater node Q source locations.

Guidance: May be called from any logic block. For a printout of the q-source data, use subroutine HNQPRT.

Restrictions: There must be at least one heater node in the submodel. Calls to this routine will reset the nodal Q for heater nodes, and should therefore only be called from VARIABLES 2, OUTPUT CALLS, and OPERATIONS DATA. Input source data will be overwritten.

Calling Sequence:

```
CALL HNQCAL('SMN')
```

6.7.2 Thermostatic Switch with Hysteresis

Subroutine Name: THRMST

Description: This subroutine provides a thermostatic switch with hysteresis. The deadband (the difference between the upper and lower temperature limit) is specified by the user.

Restrictions and Guidance: A unique user constant or other real Fortran variable must be defined for each thermostat. The initial value of the variable (0.0 or 1.0) must be initially specified, perhaps in the constants or user data block. The control constant DTMPCA (maximum allowed diffusion node temperature change) for the submodel containing node # would be some fraction of TLOW-THIGH if overshoot is an issue.

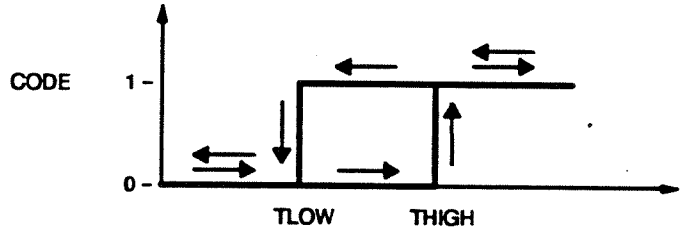
Calling Sequence:

```
M      CALL THRMST (TSEN,TLOW,THIGH,CODE)
```

where:

TSEN	sensed temperature, usually Tn where n is an actual node number
TLOW	lower temperature limit
THIGH	upper temperature limit
CODE	user "constant" or other variable unique to this thermostat

Example:



The above figure depicts the operation of the thermostatic switch. For TSEN values less than TLOW, CODE returns with a value of zero: for TSEN values greater than THIGH, CODE returns with a value of 1.0. For values between TLOW and THIGH, CODE may have a value of either 0.0 or 1.0.

The switch value CODE may be used in an F or M type statement to directly apply the heating or cooling rates. As an example of a thermostatic switch problem a typical cooling application is as follows: a thermostat is designed to switch on a cooling system rated at 200 BTU/hr at temperatures of 74 degrees and above. The cooling system remains on until 70 degrees is reached. The control thermostat is located at node 10. The cooling load is applied to source location 5. User constant 24 is assigned as the switch location:

```
CALL THRMST (CAB.T10,70.0,74.0,CAB.XK24)
```

To apply the cooling load, the following statement may be used after the above call (recall that summations are required for source values because they are reset and reconstructed each solution step):

```
Q5 = Q5 + CAB.XK24*(-200.0)
```

A typical heating application is as follows: a thermostat is designed to switch on a heater rated at 100 BTU/hr at 70 degrees and below. The heater remains on until 75 degrees is reached. The control thermostat is located at node CMP12.5. The heater load of 100 BTU/hr has been input in user constant CMP12.XK18. User constant CMP12.XK36 is assigned as the switch location.

```
CALL THRMST (CMP12.T5,70.0,75.0,CMP12.XK36)
```

To apply the heating load, the following statement may be used after the above call:

```
Q5 = Q5 + (1.0 - CMP12.XK36) * CMP12.XK18
```

6.7.3 Generalized Heater Switching

Subroutine Name: HEATER

Description: This subroutine simulates an electrical heater with a control system which turns the heater on when the sensor temperature falls below the "heater on" temperature TON, and turns the heater off when the sensor rises above the heater off temperature, TOFF. When the heater is on, the input Q value is added to the Q location specified by the user. When the heater is off, no heat is added. HEATER may be used for a fluid lump if it is called in FLOGIC 0 and QDOT has been zeroed before each call.

Restrictions: The use of such on/off control logic may cause convergence problems in steady-state solutions. If this happens, consider isolating the node in a separate submodel and either (1) calling DRPMOD during steady-states and calling ADDMOD before beginning transients, or (2) using a separate submodel with a heater (or boundary) node in steady-states, and rebuilding with the appropriate node and call to HEATER before starting a transient, replacing the temperature of the replacement node with that of the unbuilt node.

Calling Sequence:

```
 CALL HEATER (TSEN, Q, QHT, TON, TOFF, ONTIME,
+ SWITCH, CODE)
```

where:

TSEN ..... the sensed temperature
TON ..... the heater on temperature
TOFF ..... the heater off temperature
QHT ..... the heater heat rate
Q ..... the location for storing the heat
ONTIME ... a real variable set by HEATER which contains the total time the heater was on. *Use the absolute value—ignore negative values.*
SWITCH ... a real variable set by HEATER containing the number of times the heater was turned on or off.
CODE ..... a real variable set by HEATER (user sets CODE for first call):
0.0 if the heater was "on" at last call,
1.0 if the heater was "off" at the last call.

Example:

```
 CALL HEATER(T10, Q10, 500.0, 60.0, 80.0,
+  OTEST, STEST, XK10)
```

6.7.4 Heat Exchanger Simulation

Subroutine Name: HXEFF

Description: This subroutine obtains the heat exchanger effectiveness either from a user constant or from a bivariate curve of effectiveness versus the flow rates on the two sides. The effectiveness thus obtained is used with the supplied flow rates, inlet temperatures and fluid properties to calculate the outlet temperatures. The user may specify a constant effectiveness by supplying a real number or may reference an array number to specify the effectiveness as a bivariate function of the two flow rates. The user supplies flow rates, specific heat values, inlet temperatures, and a location for the outlet temperatures for each of the two sides. The specific heat values may be supplied as a temperature dependent curve or a constant value may be supplied. The user also identifies enthalpy curves for each side which may be generated from the specific heat curve with user subroutine CRVINT.

Restrictions: HXEFF should be called in the VARIABLES 1 block. The value for EFF, the first argument must never be zero. TOUT1 and TOUT2 must be boundary nodes. *This routine is not related to FLUINT network modeling, but may be used in conjunction with it.*

Calling Sequence:

```
 CALL HXEFF (EFF, W1, W2, CP1, CP2, T1N1, T1N2,
+ TOUT1, TOUT2, H1, H2)
```

where:

EFF	(1) the effectiveness if real, (2) a curve number of a bivariate curve of effectiveness versus W1 and W2 if an array
W1/2	the flow rates for side 1 and 2 respectively
CP1/2	the specific heat value for side 1 and side 2 fluid respectively. Constant values may be input or arrays may be used for temperature dependent properties
TIN1/2 ...	inlet temperatures - Usually T(IN1) and T(IN2) where IN1 and IN2 are the inlet nodes on side 1 and side 2
TOUT1/2 ..	the outlet temperature locations sides 1 and 2 where the calculated values will be returned. Must be boundary nodes
H1/2	arrays containing enthalpy vs. temperature for sides 1 and 2 respectively

Subroutine Name: HXCNT

Description: This subroutine calculates the heat exchanger effectiveness using the relationships for a counter flow type exchanger. The value of UA used in the calculations may be specified as a constant by supplying a real number or it may be specified as a bivariate function

of the two flow rates by referencing an array number. The user supplies flow rates, specific heat values, inlet temperature and a location for the outlet temperatures for each of the two sides. The specific heat values may be supplied as a temperature dependent curve or a constant value may be supplied. The user also identifies enthalpy curves for each side which may be generated from the specific heat curve with user subroutine CRVINT.

Restrictions: HXCNT should be called in the VARIABLES 1 block. The value of UA, the first argument, must never be zero. TOUT1 and TOUT2 must be boundary nodes. *This routine is not related to FLUINT network modeling, but may be used in conjunction with it.*

Calling Sequence:

```
 CALL HXCNT(UA, W1, W2, CP1, CP2, TIN1, TIN2,
+ TOUT1, TOUT2, H1, H2)
```

where:

UA	(1) the heat exchanger conductance if real, (2) a curve number of a bivariate curve of conductance versus W1 and W2 if an array
W1/2	the flow rates for side 1 and 2 respectively
CP1/2	the specific heat value for side 1 and side 2 fluid respectively. Constant values may be input or arrays may be used for temperature dependent properties
TIN1/2 ...	inlet temperatures - Usually T(IN1) and T(IN2) where IN1 and IN2 are the inlet nodes on side 1 and side 2
TOUT1/2 ..	the outlet temperature locations sides 1 and 2 where the calculated values will be returned. Must be boundary nodes
H1/2	arrays containing enthalpy vs. temperature for sides 1 and 2 respectively

Subroutine Name: HXCOND

Description: This subroutine performs thermal analysis of a condensing heat exchanger. The effectiveness may either be supplied as a constant or as a trivariate function of humidity, flow rate of the gas, and flow rate of the coolant. CRVINT may be used to integrate the specific heat curves to produce the enthalpy curves.

Restrictions: HXCOND should be called in the VARIABLES 1 block. The value for EFF, the first argument, must never be zero. TGOUT and TCOUT must be boundary nodes. *This routine is not related to FLUINT network modeling, but may be used in conjunction with it.*

Calling Sequence:

```
 CALL HXCOND(EFF,WG,WC,NHG,NHC,
+ TGIN,TCIN,PSIIN,P,XLAM,XMIMO,PSIOUT,
+ WL,TGOUT,TCOUT)
```

where:

EFF ..... (1) the effectiveness if real, (2) a curve number of a trivariate curve of effectiveness versus PSIIN, WG, and WC
WG ..... the flow rate of the gas
WC ..... the flow rate of the coolant
NHG ..... the enthalpy curve for the gas
NHC ..... the enthalpy curve for the coolant
TGIN ..... the temperature of the incoming gas
TCIN ..... the temperature of the incoming coolant
PSIIN .... the humidity of the incoming gas
P ..... the total gas pressure
XLAM ..... the latent heat of vaporization
XMIMO .... the molecular weight ratio Mv/Mo
PSIOUT ... the outlet humidity
WL ..... the flow rate of the liquid
TGOUT .... the temperature of the outgoing gas
TCOUT .... the temperature of the outgoing coolant

Subroutine Name: HXCROS

Description: This subroutine calculates the heat exchanger effectiveness using the relationships for a cross flow type exchanger. The value of UA used in the calculations may be specified as a constant by supplying a real number or it may be specified as a bivariate function of the two flow rates by referencing an array number. Any one of the following four types of cross flow exchangers may be analyzed.

1. Both streams unmixed
2. Both streams mixed
3. Stream with smallest MCp product unmixed
4. Stream with largest MCp product unmixed

The type is specified by the 10th argument in the call statement. The user supplies flow rates, specific heat values, inlet temperatures and a location for the outlet temperatures for both sides. The specific heat values may be supplied as a temperature dependent curve or a constant value may be supplied. The user also identifies enthalpy curves for each side which may be generated from the specific heat curve with user subroutine CRVINT.

Restrictions: HXCROS should be called in the VARIABLES 1 block. The value for UA, the first argument, must never be zero. TOUT1 and TOUT2 must be boundary nodes. *This routine is not related to FLUINT network modeling, but may be used in conjunction with it.*

Calling Sequence:

```
 CALL HXCROS (UA,W1,W2,CP1,CP2,TIN1,TIN2,
+ TOUT1,TOUT2,K,H1,H2)
```

where:

UA ..... (1) the heat exchanger conductance if real, (2) a curve number of a bivariate curve of conductance versus W1 and W2 if an array

W1/2 ..... the flow rates for side 1 and 2 respectively

CP1/2 .... the specific heat value for side 1 and side 2 fluid respectively. Constant values may be input or arrays may be used for temperature dependent properties

TIN1/2 ... inlet temperatures - Usually T(IN1) and T(IN2) where IN1 and IN2 are the inlet nodes on side 1 and side 2

TOUT1/2 .. the outlet temperature locations sides 1 and 2 where the calculated values will be returned. Must be boundary nodes

K ..... the code specifying type of cross flow exchanger:
Both streams unmixed: K = 1
Both streams mixed: K = 2
Stream with small WCp unmixed: K = 3
Stream with large WCp unmixed: K = 4

H1/2 ..... arrays containing enthalpy vs. temperature for sides 1 and 2 respectively

Subroutine Name: HXPAR

Description: This subroutine calculates the heat exchanger effectiveness using the relationships for a parallel flow type exchanger. The value of UA used in the calculations may be specified as a constant by supplying a real number or it may be specified as a bivariate function of the two flow rates by referencing an array. The user supplies flow rates, specific heat values, inlet temperatures and a location for the outlet temperatures for each of the two sides. The specific heat values may be supplied as a temperature dependent curve or a constant value may be supplied as a temperature dependent curve or a constant value may be supplied. The user also identifies enthalpy curves for each side which may be generated from the specific heat curve with user subroutine CRVINT.

Restrictions: HXPAR should be called in the VARIABLES 1 block. The value of UA, the first argument, must never be zero. TOUT1 and TOUT2 must be boundary temperatures. *This routine is not related to FLUINT network modeling, but may be used in conjunction with it.*

Calling Sequence:

```
 CALL HXPAR(UA,W1,W2,CP1,CP2,TIN1,TIN2,TOUT1,
+ TOUT2,H1,H2)
```

where:

UA	(1) the heat exchanger conductance if real, (2) a curve number of a bivariate curve of conductance versus W1 and W2 if an array
W1/2	the flow rates for side 1 and 2 respectively
CP1/2	the specific heat value for side 1 and side 2 fluid respectively. Constant values may be input or arrays may be used for temperature dependent properties
TIN1/2 ...	inlet temperatures - Usually T(IN1) and T(IN2) where IN1 and IN2 are the inlet nodes on side 1 and side 2
TOUT1/2 ..	the outlet temperature locations sides 1 and 2 where the calculated values will be returned. Must be boundary nodes
H1/2	arrays containing enthalpy vs. temperature for sides 1 and 2 respectively

6.7.5 Material Conductance and Capacitance

Subroutine Name: GADDi

Description: The GADD family of subroutines will combine materials and conduction paths to form a single node or conductor. They calculate the effective thermal capacitor or conductor value for the given material network. There are 22 conductor network cases and capacity for up to ten materials in a single node. These cases are shown in Figure 6–1.

To use the routines, the standard 3-blank or GEN options should be used to define the composite node or conductor. Then a single call to a GADD subroutine will evaluate the composite conductor or capacitor value.

Restrictions: GADDi should be called in VARIABLES 1.

Calling Sequence:

M CALL GADDi(G,NCASE,TA,TB,A_1,F_1,A_2,F_2...,A_i,F_i)

where:

i	Number of conductor elements, or capacitor materials in the network (integer values 2 to 10)
G	The conductor or capacitor being evaluated
NCASE	The conductor or capacitor network case from the following figure
TA, TB	Temperatures used for determining temperature dependent properties. The subroutine uses the average of TA and TB to interpolate. Input 0.0 for both if the material properties are constant. For conductors, TA and TB are the temperatures of the two nodes connected by the composite conductor. For capacitors, TA and TB are both equal to the node temperature.
A_i	Material property array of the i^{th} element in the network. Standard doublet array of property versus temperature. For constant properties input a zero.
F_i	Multiplying factor for corresponding interpolated A_i value

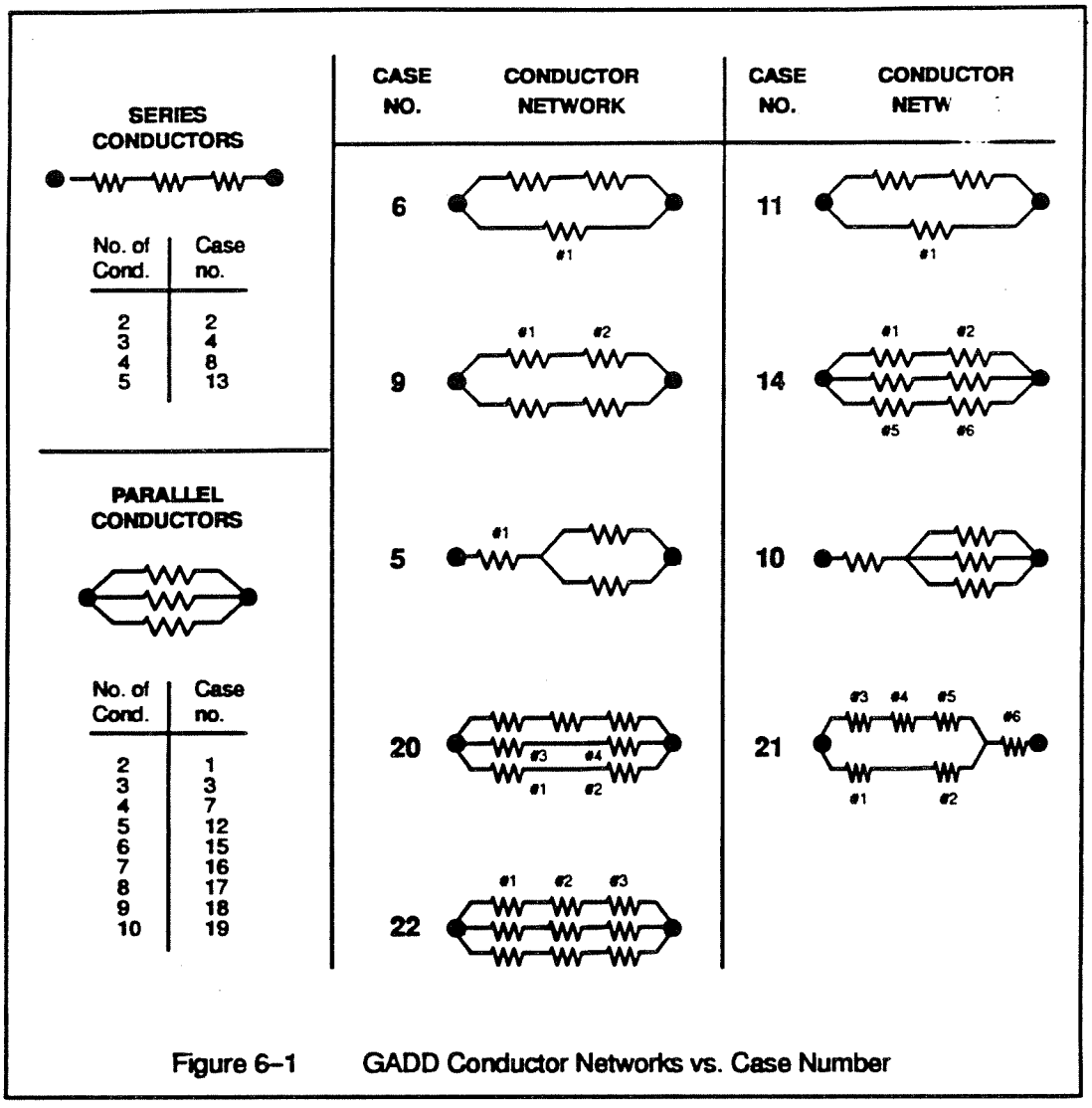


Figure 6–1 GADD Conductor Networks vs. Case Number

6.8 Arithmetic Subroutines

6.8.1 Add Two Arrays

Subroutine Name: ADDARY

Description: This subroutine will add the corresponding elements of two specified length arrays to form a third array.

Restrictions and Guidance: All data values to be operated on must be floating point numbers. The array length N must be an integer. The answer array may be overlaid into one of the input arrays.

Calling Sequence:

```
CALL ADDARY (N,AI1(DV),AI2(DV),AO(DV))
```

where:

N ..... integer size of arrays
AIn ..... input arrays
AO ..... output array

6.8.2 Add a Constant to Elements of an Array

Subroutine Name: ARYADD

Description: This subroutine will add a constant value to every element of an array to form a new array.

Restrictions and Guidance: All data values to be operated on must be floating point numbers. The array length N must be an integer. The answer array may be overlaid into one of the input arrays.

Calling Sequence:

```
CALL ARYADD (N,AI(DV),C,AO(DV))
```

where:

N ..... integer size of arrays
AI ..... input array
C ..... a floating point constant
AO ..... output array

6.8.3 Divide Corresponding Elements of Arrays

Subroutine Name: DIVARY

Description: This subroutine will divide the corresponding elements of one array into another array to form a third array.

Restrictions and Guidance: All data values to be operated on must be floating point numbers. The array length N must be an integer. The answer array may be overlaid into one of the input arrays.

Calling Sequence:

```
CALL DIVARY (N,AI1(DV),AI2(DV),AO(DV))
```

where:

N ..... integer size of arrays
AIn ..... input arrays
AO ..... output array

6.8.4 Divide a Constant into Elements of an Array

Subroutine Name: ARYDIV

Description: This subroutine will divide a constant value into every element of an array to form a new array.

Restrictions and Guidance: All data values to be operated on must be floating point numbers. The array length N must be an integer. The answer array may be overlaid into one of the input arrays.

Calling Sequence:

```
CALL ARYDIV (N,AI(DV),C,AO(DV))
```

where:

N ..... integer size of arrays
AI ..... input array
C ..... a floating point constant
AO ..... output array

6.8.5 Multiply Two Arrays

Subroutine Name: MPYARY

Description: This subroutine will multiply the corresponding elements of two specified length arrays to form a third array.

Restriction and Guidance: All data values to be operated on must be floating point numbers. The array length N must be an integer. The answer array may be overlaid into one of the input arrays.

Calling Sequence:

```
CALL MPYARY (N,AI1(DV),AI2(DV),AO(DV))
```

where:

N ..... integer size of arrays
AIn ..... input arrays
AO ..... output array

6.8.6 Multiply the Elements of an Array by a Constant

Subroutine Name: ARYMPY

Description: This subroutine will multiply every element of an array by a constant value to form a new array.

Restrictions and Guidance: All data values to be operated on must be floating point numbers. The array length N must be an integer. The answer array may be overlaid into one of the input arrays.

Calling Sequence:

```
CALL ARYMPY (N,AI(DV),C,AO(DV))
```

where:

N ..... integer size of arrays
AI ..... input array
C ..... a floating point constant
AO ..... output array

6.8.7 Subtract Corresponding Elements Of Arrays

Subroutine Name: SUBARY

Description: This subroutine will subtract the corresponding elements of one array from another array to form a third array.

Restrictions and Guidance: All data values to be operated on must be floating point numbers. The array length N must be an integer. The answer array may be overlaid into one of the input arrays.

Calling Sequence:

```
CALL SUBARY (N,AI1(DV),AI2(DV),AO(DV))
```

where AO(DV) = AI1(DV) - AI2(DV) and:

```
N ..... integer size of arrays
AIn ..... input arrays
C ..... a floating point constant
AO ..... output array
```

6.8.8 Subtract a Constant from Elements of an Array

Subroutine Name: ARYSUB

Description: This subroutine will subtract a constant value from every element of an array to form a new array.

Restrictions and Guidance: All data values to be operated on must be floating point numbers. The array length N must be an integer. The answer array may be overlaid into one of the input arrays.

Calling Sequence:

```
CALL ARYSUB (N,AI(DV),C,AO(DV))
```

where AO(DV) = AI1(DV) - C and:

```
N ..... integer size of arrays
AI ..... input array
C ..... a floating point constant
AO ..... output array
```

6.8.9 Sum an Array of Floating Point Values

Subroutine Name: SUMARY

Description: This subroutine will sum the specified elements of an array.

Restrictions and Guidance: All data values to be operated on must be floating point numbers. The array length N must be an integer.

Calling Sequence:

```
CALL SUMARY (N,AI(DV),C)
```

where:

N ..... integer size of array
AI ..... input array
C ..... a floating point output sum

6.8.10 Invert Array Elements

Subroutine Name: ARYINV

Description: This subroutine will invert the specified elements of an array in place.

Restrictions and Guidance: All data values to be operated on must be floating point numbers. The array length N must be an integer. The answer array will be overlaid into the input array.

Calling Sequence:

```
CALL ARYINV (N,AIO(DV))
```

where:

N ..... integer size of array
AIO ..... input/output array

6.8.11 Divide an Array Into a Constant

Subroutine Name: ARINDV

Description: This subroutine will divide every element of an array into a constant value to form a new array.

Restrictions and Guidance: All data values to be operated on must be floating point numbers. The array length N must be an integer. The answer array may be overlaid into one of the input arrays.

Calling Sequence:

```
CALL ARINDV (N,AI(DV),C,AO(DV))
```

where AO(DV) = C/AI(DV) and:

N ..... integer size of arrays
AI ..... input array
C ..... a floating point constant
AO ..... output array

6.8.12 Inverse of Sum of Inverses

Subroutine Name: ADARIN

Description: This subroutine calculates one over the sum of inverses of an array of values. This subroutine is useful for calculating the effective conductance of series conductors.

Restrictions and Guidance: All data values to be operated on must be floating point numbers. The array length N must be an integer.

Calling Sequence:

```
CALL ADARIN (N,AI(DV),C)
```

where:

N ..... integer size of array
AI ..... input array
C ..... a floating point output

6.8.13 Array Difference and Multiplication

Subroutine Name: QFORCE

Description: This subroutine multiplies an array by the difference between two arrays to make a new array.

Restrictions and Guidance: All data values to be operated on must be floating point numbers. The array length N must be an integer. The answer array may be overlaid into one of the input arrays.

Calling Sequence:

```
CALL QFORCE (N,AI1(DV),AI2(DV),AI3(DV),AO(DV))
```

where AI4(DV) = AI3(DV) * (AI1(DV) - AI2(DV)) and:

N ..... integer size of arrays
AIn ..... input arrays
AO ..... output array

6.8.14 Floating Difference and Multiply

Subroutine Name: QMETER

Description: This subroutine multiplies a floating point number by the difference of two other floating point numbers.

Restrictions and Guidance: All data values to be operated on must be floating point numbers. The answer may be overlaid into one of the input variables.

Calling Sequence:

```
CALL QMETER (C,D,B,A)
```

where A = B * (C - D) and A through D are floating point constants or variables. A must be a variable.

6.8.15 Array Difference and Multiply

Subroutine Name: QMTRI

Description: This subroutine multiplies an array by the difference between adjacent elements of another array to make a new array.

Restrictions and Guidance: All data values to be operated on must be floating point numbers. The array length N must be an integer. The answer array may be overlaid into one of the input arrays.

Calling Sequence:

```
CALL QMTI (N,AI1(DV),AI2(DV),AO(DV))
```

where AO(DV) = AI2(DV) * (AI1(DV) - AI1(DV+1)) and:

```
N ..... integer size of arrays
AIn ..... input arrays
AO ..... output array
```

6.8.16 Floating Point Calculation

Subroutine Name: RDTNQS

Description: This subroutine adds 460.0 to two floating point numbers, takes each to the 4th power, does a difference and multiplies this by a third floating point number

Restrictions and Guidance: All data values to be operated on must be floating point numbers.

Calling Sequence:

```
CALL RDTNQS (D,C,B,A)
```

where A = B * ((C+460.0)**4 - (D+460.0)**4) and A through D are floating point constants or variables. A must be a variable.

6.9 FLUINT Subroutines

This section describes the utility, property, correlation, simulation, and output routines that are available to the FLUINT user.

6.9.1 FLUINT Utility Subroutines

This section describes the following utility routines:

CHGLMP .... Change lump thermodynamic state
CHGVOL .... Change tank volume
CHGSUC .... Change path suc[illegible] status
GOHOMO ... Force twinned p[illegible] to stay in homogenous mode
GOSLIP ..... Undoes GOHOMO actions: enables slip mode
HLDLMP .... Hold lump thermodynamic state
HTRLMP .... Hold lump enthalpy (temperature or quality)
RELLMP .... Release HLDLMP and HTRLMP actions
INTLMP .... Return internal lump sequence number
INTPAT ..... Return internal path sequence number
INTTIE ..... Return internal tie sequence number
MFLTRN .... Return fluid submodel sequence number
XTRACT .... Return effective path suction quality
SPRIME ..... Force a capillary device to stay primed
PUTTIE ..... Move a tie from one lump and/or node to another

Subroutine Name: CHGLMP

This routine can be used to make instantaneous changes in lump thermodynamic states. A state is completely described by TL, PL, and XL. With this routine, any of these three quantities may be changed while any other is held constant. For example, PL can be changed while holding either TL or XL constant. Only the values being changed or held can be guaranteed, the third value may or may not change. For example, if the current state is subcooled and PL is changed with XL held constant at 0.0, then TL will be the smaller of the current value and the saturation pressure corresponding to the new value of PL.

This routine is normally used to make step changes in pressure or temperature in plena, but may be used to change quality and may be used on any lump. Note that changes to a tank's state will affect its mass and energy, but not its volume (even if pressure changes and the tank is compliant).

Restrictions: *Do not use within FLOGIC 1 blocks.* The desired thermodynamic state must be within the range of fluid properties. PL and TL are assumed to be in the user's unit system as selected using ABSZRO and PATMOS. Input errors result in no changes made.

Calling Sequence:

```
CALL CHGLMP (MODNAM, LUMPNO, VARY, VALUE, HOLD)
```

where:

MODNAM ...	fluid submodel name containing lump to be changed, input in single quotes such as 'LOOP'
LUMPNO ...	identifier of lump to be changed, such as 100 or 352, etc.
VARY	property to vary: 'PL', 'TL', or 'XL'
VALUE	new value for property named in VARY (single precision even if PL is being changed)
HOLD	property to hold constant: 'PL', 'TL', or 'XL'. Cannot be same as VARY

Example:

```
C CHANGE THE TEMP OF LUMP 123 TO 300 DEGREES,
C HOLDING PRESSURE CONSTANT (QUALITY MIGHT THEREFORE CHANGE)
     CALL CHGLMP ('COOLER', 123, 'TL', 300.0, 'PL')
```

Caution: CHGLMP is the singular exception to the rule that pressures are always double precision variables. The data value or variable type for VALUE must always be single precision even if a pressure is being changed. This conflict is caused by the fact that TL and XL can be changed by the same routine, and those quantities are single precision.

Subroutine Name: CHGVOL

Description: This routine can be used to make changes in tank volumes. Intensive thermodynamic properties (temperature, pressure, etc.) are conserved, so overall mass and energy will be increased or decreased according to the ratio of the volumes. This routine can be used before or after CHGLMP to adjust tank states as desired.

Restrictions: Do not use within FLOGIC 1 blocks. The new volume must be a positive number. Input errors result in no changes made.

Calling Sequence:

```
CALL CHGVOL (MODNAM, NTANK, VNEW)
```

where:

MODNAM ... fluid submodel name containing tank to be changed, input in single quotes such as 'LOOP'
NTANK .... identifier of tank to be changed, such as 100 or 352, etc.
VNEW ..... new volume for tank (ft3 or m3)

Example:

```
CALL CHGVOL ('FLU', 10300, 1.0E-4)
```

Subroutine Name: CHGSUC

Description: This routine can be used to make changes in the path phase suction status.

Restrictions: *Do not use within FLOGIC 1 blocks.* Input errors result in no changes made.

Calling Sequence:

```
CALL CHGSUC (MODNAM, NPATH, STAT)
```

where:

MODNAM ... fluid submodel name containing path to be changed, input in single quotes such as 'LOOP'

NPATH .... identifier of path to be changed, such as 100 or 352, etc.

STAT ..... new value of status flag:

- 'NORM' .. Normal, extract both phases
- 'LS' ..... Extract only liquid from defined upstream lump
- 'VS' ..... Extract only vapor from defined upstream lump
- 'RLS' .... Extract only liquid from defined downstream lump
- 'RVS' .... Extract only vapor from defined downstream lump
- 'DLS' .... Extract only liquid from either end
- 'DVS' .... Extract only vapor from either end

Example:

```
CALL CHGSUC ('COOLER', 123, 'NORM')
```

Subroutine Name: GOHOMO

Description: This routine is used to force twinned tubes and STUBE connectors to behave homogeneously, or to restrain them. If the twins are currently in the slip flow mode, their flowrates will be summed into the primary path, the flowrate of the (now ignored) secondary path will be set to zero, and the phase suction status will be reset to NORM. If the paths are currently in the homogeneous mode, they will remain in that mode until GOSLIP is called to enable the slip mode.

GOHOMO can be used to restrain a single path pair, or to restrain all twinned paths in the named submodel.

Restrictions: If the ID of the input primary path is nonzero, it must be a twinned path or the program will abort. (It does not have to be currently in the slip flow mode; GOHOMO can be repeated on the same path without error.) Do not use in FLOGIC 1.

Calling Sequence:

```
CALL GOHOMO (MOD, K)
```

where:

MOD ..... fluid submodel name, in single quotes
K ..... ID of primary path in MOD to be restrained to homogeneous flow. If zero, all twinned paths in the submodel will be restrained

Examples:

```
C
C       RESTRAINS (MAKES STAY HOMOG.) PATH 101 IN FRED:
C
         CALL GOHOMO ('FRED', 101)
C
C       RESTRAINS ALL TWINNED PATHS IN WILMA:
C
         CALL GOHOMO ('WILMA', 0)
```

Subroutine Name: GOSLIP

Description: This routine is used to reverse the effects of a prior call to GOHOMO, enabling twinned paths to resume slip flow simulation *if and when they are able.* (Slip flow does not exist in the limits of very high and very low void fractions.)

GOSLIP can be used to release a single path pair, or to release all twinned paths in the named submodel.

Restrictions: If the ID of the input primary path is nonzero, it must be a twinned path or the program will abort. (It does not have to be currently in the homogeneous mode; GOSLIP can be repeated on the same path without error.) Do not use in FLOGIC 1.

Calling Sequence:

```
CALL SLIP (MOD, K)
```

where:

MOD ..... fluid submodel name, in single quotes

K ..... ID of primary path in MOD to be released to resume slip flow. If zero, all twinned paths in the submodel will be released

Examples:

```
C
C       PATH 101 IN FRED WILL RESUME SLIP FLOW WHEN ABLE:
C
         CALL GOSLIP ('FRED', 101)
C
C       RELEASES ALL TWINNED PATHS IN WILMA:
C
         CALL GOSLIP ('WILMA', 0)
```

Subroutine Name: HLDLMP

Description: This routine is used to make a tank or junction act temporarily as a plenum or boundary. In other words, it "holds" the lump in a reference state. This action will speed the solution process and may be used as part of control logic. (See also RELLMP and HTRLMP.)

HLDLMP *must* be called for at least one lump prior to a FASTIC run if no plena are present. However, it may be called at any time in any logic block except FLOGIC 1. HLDLMP has no affect on plena. The output routine LMXTAB describes the status of any holds placed on lumps.

Restrictions: *Do not use within FLOGIC 1 blocks.* Input errors result in no holds placed. HTRLMP overrides previous HLDLMP calls and vice versa. RELLMP releases all previous holds. CHGLMP may be used concurrently.

Calling Sequence:

```
CALL HLDLMP (MODNAM, LUMPNO)
```

where:

MODNAM ... fluid submodel name containing lump to be held, input in single quotes
LUMPNO ... identifier of lump to be held

Example:

```
CALL HLDLMP ('TMS', 100)
```

Subroutine Name: HTRLMP

Description: This routine is used to hold the enthalpy of a junction, making it a "heater junction." Constant enthalpy means nearly constant temperature or quality as long as pressure does not change significantly. CHGLMP can be used to adjust the enthalpy (indirectly) to maintain constant temperatures or qualities in case pressure *does* change significantly. (See also RELLMP.) The output routine LMXTAB describes the status of any holds placed on lumps.

The power required to maintain the desired enthalpy is calculated and placed in the junction QDOT cell. In other words, instead of calculating the enthalpy given the QDOT, the QDOT is calculated given the enthalpy. *If a heater junction is tied to a node, the QDOT required to maintain the junction at the current enthalpy is assumed to come from that node.* This does not represent any real heat transfer process, but is consistent with the operation of the rest of the program. If the junction is tied to more than one node, then the required QDOT is distributed amongst the nodes according to the relative "strengths" of each tie based on the QTIEs that would have existed without the HTRLMP action.

As the name implies, heater junctions may be used like heater nodes for sizing or simulating idealized heaters. However, because they behave differently from any other network element, they can be used for many modeling purposes such as temperature or quality control systems, oversized or perfect heat exchangers, capillary pumps (by holding at saturated vapor and adding an MFRSET connector), etc. More inventive modeling tricks are sure to arise. Refer to Sample Problems E and F for examples of HTRLMP usage.

The user will recall that tanks are treated like junctions in every way within FASTIC. Thus, the enthalpy of tanks may also be held in FASTIC, but such holds are ignored in other solution routines. Effectively, calls to other solution routines temporarily release the hold on the tank, but the hold is still in place for any later FASTIC calls until RELLMP is called.

Caution: After releasing an untied heater lump, the last applicable QDOT value will remain in effect as a constant unless the user overrides this value. This is especially important for constant enthalpy tanks in FASTIC since they are considered "released" in subsequent calls to other solution routines.

Restrictions: Do not use within FLOGIC 1 blocks. Input errors result in no holds placed. HLDLMP overrides previous HTRLMP calls and vice versa. RELLMP releases all holds.

Calling Sequence:

```
CALL HTRLMP (MODNAM, LUMPNO)
```

where:

MODNAM ... fluid submodel name containing lump to be held, input in single quotes
LUMPNO ... identifier of lump to whose enthalpy is to be held

Example:

```
CALL HTRLMP ('TMS', 100)
```

Subroutine Name: RELLMP

Description: This routine releases a tank or junction from holds placed by HLDLMP or HTRLMP, returning it to normal behavior. The output routine LMXTAB describes the status of any holds placed on lumps.

Note that returning a tank or junction to normal status from a HLDLMP hold immediately forces a mass and energy balance where none was required before. Small time steps and/or many iterations may result until the changes are absorbed.

Restrictions: *Do not use within FLOGIC 1 blocks.* Input errors result in no releases made.

Calling Sequence:

```
CALL RELLMP (MODNAM, LUMPNO)
```

where:

MODNAM ... fluid submodel name containing lump to be released, input in single quotes
LUMPNO ... identifier of lump to be released

Example:

```
CALL RELLMP ('TMS', 100)
```

Function Name: INTLMP

Description: This routine provides the user with the internal location pointer for a given lump. It is analogous to routine NODTRN. This sequence number can be used to directly address the arrays TL, PL, HL, DL, XL, CX, CY, CZ, and QDOT using F-type statements. The arrays VOL, VDOT, and COMP can similarly be addressed for tanks only.

Guidance: Note that lumps are stored internally in the order: tanks, junctions, and then plena. Within each category the lumps are stored by submodel, and within each submodel they are stored according to the order in which they were input.

Input errors will result in a program abort.

Calling Sequence:

```
LSEQ = INTLMP (MODNAM, LUMPNO)
```

where:

LSEQ ..... internal sequence number of lump (program designation)
MODNAM ... fluid submodel name containing lump, input in single quotes
LUMPNO ... identifier of lump (user designation)

Examples:

```
      L = INTLMP ('TMS', 100)
      QTEST = WALL.G10 *
F     + (TL(INTLMP('DERF',332)) - 10.0)
```

Function Name: INTPAT

Description: This routine provides the user with the internal cell number for a given path. This sequence number can be used to directly address the arrays FR, TLEN, DH, AF, FC, FPOW, AC, HC, UPF, DUPI, DUPJ, and IPDC using F-type statements.

Guidance: Note that paths are stored internally in the order tubes and then connectors. Within each category the paths are stored by submodel, and within each submodel they are stored according to the order in which they were input.

Input errors will result in a program abort.

Calling Sequence:

```
KSEQ = INTPAT (MODNAM, NPATH)
```

where:

KSEQ ..... internal sequence number of path (program designation)
MODNAM ... fluid submodel name containing path, input in single quotes
NPATH .... identifier of path (user designation)

Example:

```
K = INTPAT ('TMS', 299)
```

Function Name: INTTIE

Description: This routine provides the user with the internal cell number for a given tie. This sequence number can be used to directly address the UA, QTIE, DUPN, and DUPL arrays using F-type statements.

Guidance: Note that ties are stored internally in the input order.

Input errors will result in a program abort.

Calling Sequence:

```
KSEQ = INTTIE (MODNAM, NTIE)
```

where:

KSEQ ..... internal sequence number of tie (program designation)
MODNAM ... fluid submodel name containing tie, input in single quotes
NTIE ..... identifier of tie (user designation)

Example:

```
K = INTTIE ('TMS', 544)
```

Function Name: MFLTRN

Description: This function returns the relative location of a fluid submodel name in the array of submodel names (as defined by the BUILDF records).

Restrictions and Guidance: This routine allows calls to many internal routines. Invalid fluid submodel names will *not* result in an abort; rather, a zero is returned. The user should include checks against returned zero values, since use of a zero array pointer will cause unexpected results.

Calling Sequence:

```
ISUB      = MFLTRN(MODNAM)
```

where:

MODNAM ... a fluid submodel name enclosed in single quotes
L ..... the returned submodel sequence number

Subroutine Name: XTRACT

This routine calculates the effective quality currently seen by a path in the current flowrate direction. This may not equal the quality of the upstream lump if phase suction options are active, *even if they are not active for this path.* Furthermore, for upstream junctions or FASTIC tanks, the effective quality might not equal 0.0 or 1.0 even if phase suction *is* active for this path. XTRACT output corresponds to the quality printed in PTHTAB under the heading "XL UPSTRM." See also Section 4.7.8.1.

Calling Sequence:

```
CALL XTRACT( XS, MODNAM, IPATH )
```

where:

XS ..... Returned value of the effective suction quality
MODNAM ... Fluid submodel name in single quotes
PATH ..... ID of path whose suction quality is to be evaluated

A value of -1.0 is returned if the model name and/or the path ID are invalid.

Subroutine Name: SPRIME

Description: This routine, *when called from FLOGIC 0,* can be used to force a capillary device or component model (CAPIL connectors, CAPPMP macros) to maintain a primed state. This is particularly useful during steady-state runs when capillary devices can easily deprime due to artificial pressure or quality fluctuations. Once deprimed, such devices tend to stay deprimed indefinitely. The user *must assume a capillary device stays primed and then verify that a viable solution can be achieved with that assumption.* Attempting to run without this routine will invariably result in a steady-state solution where the device is deprimed. This is a valid answer, but not always desirable. See also Section 4.7.8.

This routine uses internal calls to CHGLMP such that the following conditions are met:

1) XL of liquid lump = 0.0

2) XL of vapor lump = 1.0

3) The pressure difference between the vapor and the liquid lumps is nonnegative and no greater than the capillary structure can maintain (two times the surface tension divided by the capillary radius). Pressure drops internal to the CAPPMP macro (which are inversely proportional to the input CFC) are neglected.

4) The vapor lump temperature is subcritical.

Messages can be printed to show the changes needed to keep the device primed. Such messages should be ignored if they occur only at the start of STDSTL or FASTIC, but should not be ignored if they persist or if they appear in a transient run. Persistent, repetitive adjustments made by SPRIME at the end of a nonconvergent steady state run mean that the device cannot stay primed under the given circumstances. The adjustments made should give some clues as to why the device won't stay primed.

Guidance: Use the NSOL flag within FLOGIC 0 blocks to determine the solution routine currently being used. For CAPIL connectors and CAPPMP macros, the values of RC, XVH, and XVL can be adjusted to make the model more resistance to deprime with or without SPRIME calls. See Section 4.7.8.

Restrictions: Do not use within FLOGIC 1 blocks. Input errors result in no changes made. SPRIME cannot account for internal pressure drops within the wick, and these drops are accounted for in CAPPMP macros. In other words, if the resistance in the wick is substantial compared to the capillary pressure gain, SPRIME cannot guarantee reprime. In these cases, the user might consider adding a degradation factor to the value of RC passed to the SPRIME routine (compared to the actual RC) to account for internal resistances.

Calling Sequence:

```
CALL SPRIME (MODNAM, LMPLIQ, LMPVAP, RC, MESS)
```

where:

MODNAM ... fluid submodel name containing device or component to be primed, input in single quotes such as 'LOOP'

LMPLIQ ... identifier of liquid lump

LMPVAP ... identifier of vapor lump

RC ..... current effective two-dimensional capillary radius (for CAPIL and CAPPMP this variable may be specified as RCn, where n is the identifier of the CAPIL connector or the first path of the CAPPMP model.) May be nonpositive, which signals an infinite pressure difference capacity.

MESS ..... Message flag. If nonzero prints a messages describing the changes needed to stay primed, otherwise no messages are printed.

Examples:

```
HEADER FLOGIC 0, LOOP
      IF(NSOL .LE. 1) CALL SPRIME ('LOOP', 10, 20, RC20,1)
      CALL SPRIME ('HOTSYS', 30, 31, RC10,1)
      CALL SPRIME ('MODELA', 100, 200, 0.0001,1)
      CALL SPRIME ('CAPILY', 15, 20, -1.0, NSOL)
```

Subroutine Name: PUTTIE

Description: This routine allows users to dynamically relocate heat transfer ties during processor execution. HTU, HTN, and HTNC ties may be moved to other lumps in the same submodel, and/or moved to any other node in any valid submodel. Ties always belong in the same submodel in which they were input, and therefore cannot be moved to a lump in other fluid submodels.

When a tie moves, it retains its identity (e.g., HTU), and its current values of UA, QTIE, DUPN, and DUPL. In fact, its internal storage location (i.e., the value returned by INTTIE) does not change. Because the duplication factors are invariant, the user should exercise caution when moving the lump from one duplicated section of the network to another. If the new node is not in the same submodel as the old node, all that is required is that the new submodel name be valid. No other rule changes are involved—the tie behaves as if it had been originally located at that node. If the new submodel is not currently built, the tie becomes adiabatic. If the new submodel is built but is dormant (via a DRPMOD call), the node is treated as if it were a boundary node. PUTTIE may be called repetitively in FLOGIC blocks even if no changes are expected; no action is taken if the designated node and lump correspond to the current location of the tie.

While clever uses are sure to arise, one immediate use of PUTTIE is in combination with nonequilibrium tanks (see NONEQ0 and NONEQ1). In these tanks and others with moving boundaries, PUTTIE can be used to move ties to new nodes (which usually represent fixed walls) as the boundaries change with time. Refer to Sample Problem F.

Restrictions: Do not use in FLOGIC 1. Do not attach additional ties to CAPPMP junctions. No action will be taken if the specified tie is not an HTU, HTN, or HTNC tie. When all ties have been removed from a lump, the QDOT will be reset to zero and may be altered by the user. Conversely, moving a tie to a lump that previously had no ties will override its QDOT value. Calls to RESTAR or RESTNC do not affect prior PUTTIE calls in the current run—*the locations of ties in previous runs are ignored as are PUTTIE calls made in those runs.*

PUTTIE internally calls INTTIE, INTLMP, and NODTRN. Hence, with the exception of zeroes used as flags (as defined below), input errors will result in a program abort.

Calling Sequence:

```
CALL PUTTIE(MODNAM, NTIE, LUMP, MODTHM, NODE)
```

where:

MODNAM ...	fluid submodel name containing tie, input in single quotes
NTIE	identifier of tie to be moved (user designation). HTU, HTN, or HTNC only.
LUMP	identifier of new lump in same submodel (user designation), or zero if the lump remains the same
MODNAM ...	thermal submodel name containing the node, input in single quotes
NODE	identifier of new node (user designation), or zero if the node remains the same

To move the tie to a new node without changing the lump, set the lump number to zero. To move the tie to a new lump without changing the node, set the node number to zero, in which case the thermal submodel input is ignored and can be anything.

Examples:

```
CALL PUTTIE('WET', 4559, 455, 'WARM', 10)
CALL PUTTIE('WILD', 59, 0, 'WOOLLY', 7)
CALL PUTTIE('WILLING', 1422, 999, ' ', 0)
```

6.9.2 Property Subroutines

This section describes the thermodynamic and transport property routines used in FLUINT that are available for direct calls by the user. These are internal routines that have been documented so as to be accessible by the user. This distinction is important because little to no error trapping or handling logic is employed within these routines to save execution costs during normal calls by the program itself. For this reason, ***users are cautioned to read this section carefully before attempting direct calls, and to exercise similar care when writing logic that employs these calls.***

Twenty fluids commonly used in the room temperature regime (approx. 150 to 600 K) are available in FLUINT library. They are referred to by their ASHRAE refrigeration number, as listed in Table C–1. In addition, the user may refer to any fluid defined in an FPROP DATA block by the fluid identification number (from 6000 to 9999) selected in that block. Fluids properties are available whether or not they are used in any fluid submodels.

Both transport (viscosities and conductivities) and thermodynamic properties (pressures, temperatures, enthalpies, densities, etc.) are available. These properties are based in two *absolute* units sets, according the values of UID, ABSZRO, and PATMOS selected in the CONTROL DATA, GLOBAL block. These unit sets are listed in Table D–1.

These routines agree with tabulated ASHRAE values for standard library fluids. Refer to Appendix C for the ranges of valid properties, which are also available using routines VTMIN, VTGMAX, VTLMAX, and VPGMAX.

The routines available and their arguments are listed in Table 6–1. For user–defined fluids, some routines will not be applicable, or will not be available for 6000 series fluids unless certain optional inputs have been provided. If routines are called for which inadequate input has been provided, a warning will be produced.

Vapor transport properties are functions of saturation temperature only for standard library fluids, even though pressure is an input argument. This argument is applicable for 6000 series fluids only.

Note that *all inputs and outputs must be in absolute units*, and that the last argument is a fluid identifier. The fluid identifier must be translated; the routine call cannot be preceded by an F in column 1. There are two ways of specifying the fluid identifier, as described below:

Identifier (Last argument)	Fluid identified
`smn.FI`	Indirect method: identifies the fluid used in submodel smn where smn is the name of a valid fluid submodel.
`FIn or FI(n)`	Direct method: identifies fluid number n, where n is a valid number from Table C–1 or from an FPROP DATA block.

Unlike any other SINDA/FLUINT variable, *all pressures must be double precision* floating point numbers. This may cause some difficulties when calling property routines from logic blocks. If the user is not using lump PL values, which are always double precision, then he must be careful to pass a double precision quantity to the property routines. (NOTE: CHGLMP is not considered a property routine and hence expects single precision inputs.) Numeric constants

Table 6-1 Property Routines

FUNCTION	OUTPUT	INPUTS
VPS	Saturation P	T
VTS	Saturation T	P
VSV	Vapor V	P,T
VH	Vapor H	P,T,V
VS	Vapor S	T,V
VDL	Liquid Density	T
VHFG	Heat Of Vaporization	P,T,V (vap.)
VQUALH	X	P,H
VVISCF	Liquid Viscosity	T
VVISCV	Vapor Viscosity	P,T
VCPF	Liquid Heat Capacity	P,T
VHLIQ	Liquid H	P,T
VTLIQ	Liquid T	P,H
VDPDT	Saturation dP/dT	P
VDPDTT	Saturation dP/dT	T
VCPV	Vapor Heat Capacity	P,T
VQUALS	X	P,S
VST	Liquid/Vapor Surface Tension	T
VCONDF	Liquid Thermal Conductivity	T
VCONDV	Vapor Thermal Conductivity	P,T
VALPHA	Vapor Volume Fraction	T,X
VSOS	Vapor Speed Of Sound	P,T
VDLC	Compressed Liq Density	P,T
VSOSF	Liquid Speed of Sound	P,T
VSOSFV	Two-phase Speed of Sound	P,T,X,M
VTMIN	Minimum temp., any phase	-
VTGMAX	Maximum vapor/gas temp.	-
VTLMAX	Maximum liquid temp.	-
VPGMAX	Maximum vapor/gas press.	-

SUBROUTINE	OUTPUT	INPUTS
VTAV1	Vapor T,V	P,S
VTAV2	Vapor T,V	P,H

Where:
P ..... Pressure (absolute)
T ..... Temperature (absolute)
X ..... Quality (Vapor To Total Mass Ratio)
H ..... Enthalpy, Intensive
S ..... Entropy, Intensive
V ..... Specific Volume
M ..... Two-pghase sound speed method (1 or 2)

must include the "D" form of the exponent to signify double precision, such as 1.0D0. Because user constants cannot be double precision, the user must use a local double precision variable and turn the DEBUG feature off using the DEBOFF command in the logic blocks or the NODEBUG command in OPTIONS DATA. Remember, the control constant PATMOS is double precision, as well as functions VPS and VPGMAX.

Examples:

```
DEBOFF                                    $ ignore locals
      DOUBLE PRECISION PRES               $ local D.P.
                .
                .
                .
      TEMP = VTS(PRES-PATMOS,LOOP.FI)     $ fluid in LOOP
      CALL VTAV2(T,V,PL33-PATMOS,
     +    HL33,FI718)                     $ water
      VDEN = 1.0/VSV(PRES-PATMOS,
     +    TEMP-ABSZRO,FI717)              $ ammonia
      VISC = VVISCF(MAIN.TL44-ABSZRO,
     +    FI9014)                         $ user fluid
      HFG  = VHFG(14.7D0-PATMOS,          $ note D.P.
     +    YY.TL99-ABSZRO,1.0/VDEN,XX.FI)
```

The PRPMAP output routine, described in Section 6.9.5, may be used to print out working fluid properties. Because it is considered an output routine, input temperatures and pressures are expected in relative units.

Refer also to Section 3.13 for information regarding alternative fluid descriptions.

VSOSFV Description—This routine calculates speed of sound in a two–phase mixture using one of two methods: Wallis' adiabatic method or Bursik's metastable method. The method to be used is signaled by the third argument, M, which is an integer: 1 for Wallis', 2 for Bursik's method.

The first method is simple and fast, but may overpredict the speed of sound by a little (typically a few percent, but can be up to 20%). It predicts speed of sound by assuming a homogeneous mixture (at the input quality) of two adiabatic phases, combining the phasic compressibilities like springs in series. Wallis also proposed another isothermal method which better predicts sound speeds in two–phase mixtures, but does not predict the right value in the limits of 100% vapor.

The second method (Bursik's, NASA TM 78810) is computationally slower, but combines the best features of Wallis' two methods: it is a better estimation at low qualities, and it predicts the correct value in the limit of 100% vapor. However, the user should note that, like almost any two–phase prediction, there is no definitive method or exact solution. In other words, either method is probably as accurate as the other, since neither take into account flow regime variations and other effects that are known to be very important.

This routine is invalid for 6000 fluids unless the VSOSFV function has been input or if the VDLC *and/or* the VSOSF functions have been input. Quality may range from 0.0 to 1.0, inclusive.

6.9.3 Heat Transfer Correlation Functions and Routines

This section describes the heat transfer correlation routines used in FLUINT that are available to the advanced user. The references for these routines and their technical descriptions are found in Appendices A and B. See also Sample Problems C and E.

This section describes the following heat transfer routines:

STDHTC .... Central call to convection correlation
DITTUS ..... Single-phase laminar and turbulent correlation
ROHSEN .... Condensation correlation (standard)
ACKERS .... Condensation correlation (alternate)
SHAH ..... Condensation correlation (alternate)
CHENNB .... Nucleate boiling correlation

Subroutine Name: STDHTC

Description: This subroutine is the standard heat transfer correlation routine set, as described in Appendix B. Given the inner wall temperature and the geometry of a pipe segment (length, flow area, and hydraulic diameter), the mass flowrate and flow quality through the segment and the thermodynamic state of the fluid (pressure, temperature, and void fraction), STDHTC calculates the heat transfer conductance UA from the inner pipe wall to the bulk fluid. DITTUS, ROHSEN, and CHENNB are called as needed to calculate the heat transfer coefficient. STDHTC assumes thermodynamic and hydrodynamic equilibrium. The area used for heat exchange is based on the wetted perimeter: 4.0*TLEN*AF/DH.

If required, the heat rate may be calculated as the product of the returned UA coefficient times the difference in temperature between the node and lump: UA*(TW-TL).

UA is estimated in the case where the temperature difference is zero (TL=TW) but the flowrate is nonzero. If the flowrate is zero, the routine will use the laminar Nusselt number as a limit. If the flowrate is zero and the fluid is two–phase, vapor properties will be used for heating and liquid properties for cooling.

Restrictions: All arguments are assumed in standard, *absolute* units according to the unit flag UID selected in global constants data. The thermodynamic state must be within the range of fluid properties, and the fluid is specified indirectly by naming the *internal sequence number* of the FLUINT submodel containing the desired fluid (see MFLTRN). Pressures are double precision.

Calling Sequence:

```
CALL STDHTC (UA,FR,DH,AF,TLEN,PL,TL,XL,ALPH,TW,MODNO)
```

where:

UA	returned heat transfer coefficient times the wetted heat transfer area 4.0*TLEN*AF/DH.
FR	mass flowrate through the segment
DH	hydraulic diameter of the segment
AF	cross-sectional (flow) area of the segment
TLEN	length of the segment
PL	absolute pressure of the fluid (double precision)
TL	absolute temperature of the fluid
XL	flow quality
ALPH	void fraction of the fluid
TW	inner wall temperature of the segment (absolute)
MODNO	Model number containing the desired fluid. If only the model name is known, use MFLTRN ('SMN') as this argument, where SMN is the submodel name.

Function Name: DITTUS

Description: This function returns the heat transfer coefficient for single phase flow based on the Dittus-Boelter correlation for turbulent flow or the isothermal circular Nusselt number (3.66) for laminar flow. Hausen's transition correlation is used for Reynold's numbers between approximately 2000 and 6400.

Restrictions: All units are assumed in standard, *absolute* form according to the unit flag UID selected in global constants data. FI must be translated; do *not* place an F in column 1 when calling this routine. Pressures are double precision.

Calling Sequence:

```
H = DITTUS (FR,DH,AF,TW,PL,TL,XL,FI)
```

where:

H	heat transfer coefficient
FR	mass flowrate through the segment (positive)
DH	hydraulic diameter of the segment
AF	cross-sectional (flow) area of the segment
TW	inner wall temperature (compared with TL only, magnitude is not important)
PL	absolute pressure of the fluid (double precision)
TL	absolute temperature of the fluid
XL	quality or phase flag, 1.0 for vapor, 0.0 for liquid
FI	fluid identifier: FIn where n is the fluid ASHRAE number or user-defined fluid number, or smn.FI where smn is the desired fluid submodel name

Function Name: ROHSEN

Description: This function returns the condensation heat transfer coefficient for two-phase flow based on Rohsenow's correlation (see Appendix B). This is the condensation correlation used in HTN and HTNC ties.

Restrictions: All units are assumed in standard, *absolute* form according to the unit flag UID selected in global constants data. FI must be translated; do *not* place an F in column 1 when calling this routine. TW must be less than TL. XL and ALPH must be between 0.0 and 1.0, exclusive. FR must be greater than zero. Pressures are double precision.

Calling Sequence:

```
H = ROHSEN (FR,DH,AF,TW,PL,TL,XL,ALPH,FI)
```

where:

H	heat transfer coefficient
FR	mass flowrate through the segment (positive)
DH	hydraulic diameter of the segment
AF	cross-sectional (flow) area of the segment
TW	inner wall temperature (must be less than TL)
PL	absolute pressure of the fluid (double precision)
TL	absolute temperature of the fluid
XL	flow quality: greater than zero, less than one
ALPH	void fraction: greater than zero, less than one
FI	fluid identifier: FIn where n is the fluid ASHRAE number or user-defined fluid number, or smn.FI where smn is the desired fluid submodel name

Function Name: ACKERS

Description: This alternate function returns the condensation heat transfer coefficient for two-phase flow based on Acker's correlation (see American Society of Heating, Refrigeration, and Air Conditioning Engineers (ASHAE) Fundamentals Handbook).

Restrictions: All units are assumed in standard, *absolute* form according to the unit flag UID selected in global constants data. FI must be translated; do *not* place an F in column 1 when calling this routine. TW must be less than TL. FR and XL must be greater than zero. Pressures are double precision.

Calling Sequence:

```
H = ACKERS (FR,DH,AF,TW,PL,TL,XL,FI)
```

where:

H ..... heat transfer coefficient
FR ..... mass flowrate through the segment (positive)
DH ..... hydraulic diameter of the segment
AF ..... cross-sectional (flow) area of the segment
TW ..... inner wall temperature (must be less than TL)
PL ..... absolute pressure of the fluid (double precision)
TL ..... absolute temperature of the fluid
XL ..... flow quality: greater than zero, less than one
FI ..... fluid identifier: FIn where n is the fluid ASHRAE number or user-defined fluid number, or smn.FI where smn is the desired fluid submodel name

Function Name: SHAH

Description: This alternate function returns the condensation heat transfer coefficient for two-phase flow based on Shah's correlation. This highly empirical correlation is extremely simplistic, but is favored by some analysts.

Restrictions: All units are assumed in standard, *absolute* form according to the unit flag UID selected in global constants data. FI must be translated; do *not* place an F in column 1 when calling this routine. TW must be less than TL. FR and XL must be greater than zero. Pressures are double precision. The critical pressure is used within this correlation.

Calling Sequence:

```
H = SHAH (FR,DH,AF,TW,PL,TL,XL,FI)
```

where:

H ..... heat transfer coefficient
FR ..... mass flowrate through the segment (positive)
DH ..... hydraulic diameter of the segment
AF ..... cross-sectional (flow) area of the segment
TW ..... inner wall temperature (must be less than TL)
PL ..... absolute pressure of the fluid (double precision)
TL ..... absolute temperature of the fluid
XL ..... flow quality: greater than zero, less than one
FI ..... fluid identifier: FIn where n is the fluid ASHRAE number or user-defined fluid number, or smn.FI where smn is the desired fluid submodel name

Function Name: CHENNB

Description: This function returns the nucleate boiling heat transfer coefficient for two-phase flow based on Chen's correlation (see Appendix B).

Restrictions: All units are assumed in standard, *absolute* form according to the unit flag UID selected in global constants data. FI must be translated; do *not* place an F in column 1 when calling this routine. TW must be greater than TL. XL and ALPH must be less than 1.0, exclusive. FR must be greater than zero. Pressures are double precision.

Calling Sequence:

```
H = CHENNB (FR,DH,AF,TW,PL,TL,XL,ALPH,FI)
```

where:

H	heat transfer coefficient
FR	mass flowrate through the segment (positive)
DH	hydraulic diameter of the segment
AF	cross-sectional (flow) area of the segment
TW	inner wall temperature (must be greater than TL)
PL	absolute pressure of the fluid (double precision)
TL	absolute temperature of the fluid
XL	flow quality: less than one
ALPH	void fraction: less than one
FI	fluid identifier: FIn where n is the fluid ASHRAE number or user-defined fluid number, or smn.FI where smn is the desired fluid submodel name

6.9.4 Pressure Drop Correlation Functions and Routines

This section describes the frictional pressure drop correlation routines used in FLUINT that are available to the user. The references for these routines are found in Appendix A.

This section describes the following pressure drop routines:

FRICT ..... Single-phase laminar and turbulent correlation (Moody Chart fit)
MCADAM ... Two-phase Frictional Pressure Gradient (McAdam's Homogeneous)
LOCMAR .... Two-phase Frictional Pressure Gradient (Lockhart-Martinelli's)
BAROC ..... Two-phase Frictional Pressure Gradient (Baroczy's)
FRIEDL ..... Two-phase Frictional Pressure Gradient (Friedel's)

Function Name: FRICT

Description: Returns the Darcy friction factor based on a Reynold's number and a wall roughness ratio. This is a curve fit to the Moody chart.

Calling Sequence:

```
F = FRICT (RE,WRF)
```

where:

F ..... Darcy friction factor
RE ..... fluid Reynold's number
WRF ..... wall roughness fraction (may be zero for smooth walls)

Subroutine Name: MCADAM

Description: This routine returns the two–phase frictional pressure gradient based on McAdam's homogenous flow correlation. See Appendix A.

Restrictions: All units are assumed in standard, absolute form according to the u- -flag UID selected in global constants data. FR must be greater than zero.

Calling Sequence:

```
 CALL MCADAM (DPZ,VISL,VISV,RHOL,RHOV,FR,XL,
+ DH,AF,WRF
```

where:

DPZ ..... returned two-phase frictional pressure gradient (pressure per unit length)
VISL ..... liquid viscosity
VISV ..... vapor viscosity
RHOL ..... liquid density
RHOV ..... vapor density
FR ..... mass flowrate (positive)
XL ..... flow quality
DH ..... hydraulic diameter
AF ..... flow area
WRF ..... wall roughness fraction

Subroutine Name: LOCMAR

Description: This routine returns the two-phase frictional pressure gradient based on the Lockhart-Martinelli correlation with curve fits by Chisolm. See Appendix A.

Restrictions: All units are assumed in standard, absolute form according to the unit flag UID selected in global constants data.

Calling Sequence:

```
CALL LOCMAR (DPZ,DPL,DPV,REL,REV)
```

where:

DPZ ..... returned two-phase frictional pressure gradient (pressure per unit length)
DPL ..... liquid pressure gradient if no vapor were present
DPV ..... vapor pressure gradient if no liquid were present
REL ..... liquid Reynolds number if no vapor were present
REV ..... vapor Reynolds number if no liquid were present

Subroutine Name: BAROC

Description: This routine returns the two-phase frictional pressure gradient based on Baroczy's with curve fits by Chisolm. See Appendix A.

Restrictions: All units are assumed in standard, absolute form according to the unit flag UID selected in global constants data. FR must be greater than zero.

Calling Sequence:

```
CALL BAROC (DPZ,DPLO,DPVO,FR,AF,XL)
```

where:

DPZ ..... returned two-phase frictional pressure gradient (pressure per unit length)
DPLO ..... pressure gradient if all mass flow were liquid
DPVO ..... pressure gradient if all mass flow were vapor
FR ..... mass flowrate (positive)
AF ..... flow area
XL ..... flow quality

Subroutine Name: FRIEDL

Description: This routine returns the two-phase frictional pressure gradient based on Friedel's correlation. See Appendix A.

Restrictions: All units are assumed in standard, *absolute* form according to the unit flag UID selected in global constants data. FR must be greater than zero. FI must be translated; do *not* place an F in column 1 when calling this routine.

Calling Sequence:

```
 CALL FRIEDL (DPZ,RHOL,RHOV,VISL,VISV,XL,
+ FLO,FVO,FR,AF,DH,TL,FI)
```

where:

DPZ ..... returned two-phase frictional pressure gradient (pressure per unit length)

RHOL ..... liquid density

RHOV ..... vapor density

VISL ..... liquid viscosity

VISV ..... vapor viscosity

FLO ..... Darcy friction factor if all mass flow were liquid

FVO ..... Darcy friction factor if all mass flow were vapor

XL ..... flow quality

FR ..... mass flowrate through the segment (positive)

AF ..... cross-sectional (flow) area of the segment

DH ..... hydraulic diameter of the segment

TL ..... absolute temperature of the fluid

FI ..... fluid identifier: FIn where n is the fluid ASHRAE number or user-defined fluid number, or smn.FI where smn is the desired fluid submodel name

6.9.5 FLUINT Output Routines

This section describes the following fluid submodel output routines:

LMPTAB .... Tabulates basic lump parameters
LMXTAB .... Tabulates extra lump parameters
TIETAB ..... Tabulates tie parameters
PTHTAB .... Tabulates path parameters
TWNTAB .... Tabulates twinned path parameters
TUBTAB .... Tabulates tube and STUBE parameters
LMPRNT .... Prints fluid lump attributes
QDPRNT .... Prints fluid lump energy flow
TKPRNT .... Prints fluid tank attributes
FRPRNT .... Prints fluid path flowrates
TBPRNT .... Prints fluid tube attributes
STPRNT .... Prints fluid STUBE connector attributes
FLOMAP .... Prints fluid network connectivities
LMPMAP .... FLOMAP for one lump
PRPMAP .... Maps working fluid properties
UDFERR .... User error routine for 6NNN fluids
YSMPWK ... YSMP workspace usage report

Subroutine Name: LMPTAB

Description: This routine tabulates a variety of lump parameters, including TL, PL, XL, DL, HL, and QDOT (listed as "HEAT RATE"). Two headings requiring further explanation are listed below.

"MASS RATE" — The rate of change of mass for a lump, or the mass influx for a plenum. This should be nearly zero for a junction unless HLDLMP has been used, and should approach zero for a tank at steady-state. Thus, this can be used as a measure of how far a lump is from steady conditions.

"ENERGY RATE" — The rate of change of energy for a lump, or the energy influx for a plenum. This should be nearly zero for a junction unless HLDLMP has been used, and should approach zero for a tank at steady-state. Thus, this can be used as a measure of how far a lump is from steady conditions.

Calling Sequence:

```
CALL LMPTAB ('SMN')
```

where:

SMN ..... a fluid submodel name (use the word 'ALL' to print all fluid submodels). Character delineators (') are required.

Subroutine Name: LMXTAB

Description: This routine tabulates lump parameters excluded from LMPTAB that are useful for model debugging purposes, including lump coordinates (CX, CY, CZ) and body force vector components, the number of paths attached to each lump, whether or not each lump has been held using HLDLMP or HTRLMP, and various tank attributes such as volume, mass, and compliance.

While LMXTAB can be called from any logic block, the general invariance of the parameters it tabulates make it most appropriate to be called from OPERATIONS DATA before and/or between solution routines.

Calling Sequence:

```
CALL LMXTAB ('SMN')
```

where:

SMN ..... a fluid submodel name (use the word 'ALL' to print all fluid submodels). Character delineators (') are required.

Subroutine Name: TIETAB

Description: This routine tabulates a variety of tie parameters, including UA, the QTIE currently impressed on the tied lump, and the names and temperatures of the tied lump and node. For HTN, HTNS, and HTNC ties, the names of the associated paths and the corresponding area fractions are also listed.

Guidance: The heat rate reported in the tie may not always be equal to the UA value times the temperature difference. The reasons for this apparent discrepancy are varied, and include internal damping and other such stability enhancing measures. In general, the discrepancy does not exist upon convergence, or is apparent only for negligibly small heat rates.

Calling Sequence:

```
CALL TIETAB ('SMN')
```

where:

SMN ..... a fluid submodel name (use the word 'ALL' to print all fluid submodels). Character delineators (') are required.

Subroutine Name: PTHTAB

Description: This routine tabulates a variety of path parameters, including upstream and downstream lumps, FR, DUPI, DUPJ and STAT. Two headings requiring further explanation are listed below.

"XL UPSTRM" — The quality of the upstream lump as viewed by this path, taking into account any phase suction action. This is the same value returned by the utility routine XTRACT.

"FLOW REGIMES OR REYNOLDS NO." — (Tubes and STUBE connectors only) If two-phase flow is present, prints the current flow regime if IPDC=6, else UNKNOWN. For single-phase flow, prints the current Reynolds number. One or more regimes or numbers may appear depending on the value of UPF and the direction of flowrate. The first column corresponds to the defined upstream end (LMP 1), and the second column to the defined downstream end (LMP 2).

When paths are twinned and active, the primary twin ID will be listed with an 'L' suffix, and the secondary twin ID with a 'V' suffix. In this case, the Reynolds number for each twin corresponds to the superficial velocity (i.e., as if this were the only phase in the duct. Otherwise, for twins in the homogeneous mode only the primary twin will be listed with a suffix of 'H' after the ID.

Calling Sequence:

```
CALL PTHTAB ('SMN')
```

where:

SMN ..... a fluid submodel name (use the word 'ALL' to print all fluid submodels). Character delineators (') are required.

Subroutine Name: TWNTAB

Description: This routine tabulates a variety of path parameters for twinned paths, including upstream and downstream lumps, FR, DUPI, DUPJ and STAT. TWNTAB is an alternate way of interpreting the same data presented for twins in PTHTAB, and is intended to be used as a companion to PTHTAB when twins are used. Three headings requiring further explanation are listed below.

"XL FLOW" — The quality of the flow through the passage shared by the pair of twins, calculated by $|FR_v|$ divided by the quantity $|FR_v| + |FR_L|$.

"FR TOTAL" — The total mass flow through the passage shared by the pair of twins, calculated as $FR_v + FR_L$.

"FLOW REGIMES OR REYNOLDS NO." — If two–phase flow is present, prints the current flow regime if IPDC=6, else UNKNOWN. For single–phase flow, prints the current Reynolds number. One or more regimes or numbers may appear depending on the value of UPF and the direction of flowrate. The first column corresponds to the defined upstream end (LMP 1), and the second column to the defined downstream end (LMP 2).

When paths are twinned and active, the primary twin ID will be listed with an 'L' suffix, and the secondary twin ID with a 'V' suffix. In this case, the flow regimes are printed for the pair if IPDC=6. (Only the upstream end will appear even if UPF=0.5, since twins internally use UPF=1.0 when active.) Otherwise, for twins in the homogeneous mode the primary twin will be listed with a suffix of 'H' after the ID, and the secondary will appear with an 'X' suffix.

Calling Sequence:

```
CALL TWNTAB ('SMN')
```

where:

SMN ..... a fluid submodel name (use the word 'ALL' to print all fluid submodels). Character delineators (') are required.

Subroutine Name: TUBTAB

Description: This routine tabulates tube and STUBE simulation parameters: DH, AF, TLEN, AC, HC, FC, FPOW, UPF, IPDC, FK, and WRF.

Calling Sequence:

```
CALL TUBTAB ('SMN')
```

where:

SMN ..... a fluid submodel name (use the word 'ALL' to print all fluid submodels). Character delineators (') are required.

Subroutine Name: LMPRNT

Description: This routine prints out a variety of lump thermodynamic state variables. A print flag controls the printing of TL, PL, XL, HL and DL for a submodel.

Restrictions: A fluid submodel name must be specified and should agree with a submodel name appearing on a BUILDF card.

Calling Sequence:

```
CALL LMPRNT ('SMN', 'PRFLG')
```

where:

SMN ..... a fluid submodel name; (use the word 'ALL' to print all fluid submodels) Character delineators (') are required.

PRFLG .... A character or combination of characters that define the variables to be printed. Valid entries are:
'T' ..... to print TL
'P' ..... to print PL
'X' ..... to print XL
'H' ..... to print HL
'D' ..... to print DL
Or the word 'ALL'. Character delineators (') are required.

Subroutine Name: QDPRNT

Description: This subroutine prints out the value of QDOT for lumps.

Restrictions: A fluid submodel name must be specified and should agree with a submodel name appearing on a BUILDF card.

Calling Sequence:

```
CALL QDPRNT ('SMN')
```

where SMN is a fluid submodel name, or use the word 'ALL' to print all fluid submodels. Character delineators (') are required.

Subroutine Name: TKPRNT

Description: This subroutine prints out the values of mass, volume, rate of change of volume (growth/shrink rate), and compliance factors for tanks.

Restrictions: A fluid submodel name must be specified and should agree with a submodel name appearing on a BUILDF card.

Calling Sequence:

```
CALL TKPRNT ('SMN')
```

where SMN is a fluid submodel name, or use the word 'ALL' to print all fluid submodels. Character delineators (') are required.

Subroutine Name: FRPRNT

Description: This subroutine prints out flow rates for paths along with path type and connector device names.

Restrictions: A fluid submodel name must be specified and should agree with a submodel name appearing on a BUILDF card.

Calling Sequence:

```
CALL FRPRNT ('SMN')
```

where SMN is a fluid submodel name or use the word 'ALL' to print all fluid submodels. Character delineators (') are required.

Subroutine Names: TBPRNT, STPRNT

Description: These subroutines print out the values of DH, AF and TLEN for tubes or STUBE connectors.

Restrictions: A fluid submodel name must be specified and should agree with a submodel name appearing on a BUILDF card.

Calling Sequence:

```
CALL TBPRNT ('SMN')
CALL STPRNT ('SMN')
```

where SMN is a fluid submodel name or use the word 'ALL' to print all fluid submodels. Character delineators (') are required.

Subroutine Name: FLOMAP

Description: This subroutine prints out the connectivity network for a fluid submodel. It prints out the mass flowrate and energy rate associated with a path (positive *into* each lump). It also prints out the temperature and quality of the adjoining lump, along with the difference in pressures (adjoining minus current). After the path information is printed for each lump FLOMAP prints out the total mass and energy storage rate for the lump. FLOMAP may be used to map a single model or all fluid submodels. If the QMAP file name is set in OPTIONS DATA, FLOMAP will write to this file. Output can optionally be routed to the OUTPUT file, although this copy can be suppressed if a QMAP file is named.

Restrictions and Guidance: Because of the volume of output, FLOMAP calls should only appear in OPERATIONS DATA and must never appear in any VARIABLES or FLOGIC block. A fluid submodel name must be specified and should agree with a submodel name appearing on a BUILDF card.

Calling Sequence:

```
CALL FLOMAP (SMN, NPRNT)
```

where:

SMN ..... a fluid submodel name in quotes or 'ALL'
NPRNT .... OUTPUT Print / No Print flag:
0: print to processor OUTPUT file
1: suppress printing to OUTPUT file (still writes to QMAP file if named in OPTIONS DATA)

Subroutine Name: LMPMAP

Description: This subroutine performs the same function as FLOMAP, but maps only one requested lump to help reduce outputs.

Calling Sequence:

```
CALL LMPMAP (SMN, LUMP, NPRNT)
```

where:

SMN ..... a fluid submodel name in quotes or 'ALL'
LUMP ..... ID of lump to be mapped
NPRNT .... OUTPUT Print / No Print flag:
0: print to processor OUTPUT file
1: suppress printing to OUTPUT file (still writes to QMAP file if named in OPTIONS DATA)

Subroutine Name: PRPMAP

Description: This output routine is available to aid in the development of a fluid description input in an FPROP DATA block. This routine should be called once from OPERATIONS DATA. It produces tabulated properties for any fluid over the valid range of temperatures and pressures. The property information is written to the OUTPUT file.

Calling Sequence:

```
CALL PRPMAP( P, T, smn.FI or FIn )
```

where:

P	Pressure, double precision. For 8000 and 9000 fluids, this is the pressure at which tables will be produced. For other fluids, this is the *maximum* pressure for which data will be produced if it is smaller than PGMAX.
T	Maximum temperature at which data will be produced if it is smaller than TGMAX (or TLMAX for 9000 fluids).
smn	submodel containing desired fluid
n	fluid ID number

Restrictions and Guidance: The smn.FI or FIn format choice is the same as that of other property routines, but that *P and T are assumed to be in nonabsolute, user units*, whereas they must be in absolute units for other property calls. All output is in the unit system convention chosen by the user-input values of UID, PATMOS, and ABSZRO.

Subroutine Name: UDFERR

Description: This simple routine is provided to allow the user to write warnings to the OUTPUT file. The primary purpose is to support 6000 series user-defined fluids. The calling sequence is:

```
CALL UDFERR(ROUT,ID,LINE)
```

where:

ROUT	Routine name where error occurred, character variable
ID	Fluid ID, integer
LINE	Status or message line, character variable.

This routine prints the following message to the output file:

```
 *** WARNING *** ILLEGAL OPERATION IN PROPERTY ROUTINE (ROUT)
(LINE); FID: (ID), TYPE: ADVANCED TWO-PHASE
```

Subroutine Name: YSMPWK

Description: This output routine is available to aid in the customizing of the YSMP (matrix inversion package) workspace allocation. YSMPWK simply reports on the actual usage (as experienced thus far in the run) of workspace by each submodel. It is called automatically by the processor in the event of an abort due to insufficient allocation, and is available for direct user calls as well.

The exact amount workspace required cannot be known beforehand, and insufficient allocation will result in an aborted run. The workspace allocation for each fluid submodel is normally calculated automatically by the preprocessor, which employs a conservative estimate. The user may override this estimate by specifying the allocation with the "NWORK=I" keyword within the HEADER FLOW DATA subblock. *Unless directed to do so by the processor abort message, the user should avoid employing this option.* Refer to the caution below.

Calling Sequence:

```
CALL YSMPWK
```

Guidance: Since the utilization of workspace is cumulative, *place the call to YSMPWK at the end of OPERATIONS DATA.* Calling this routine from other locations may result in lower than actual reported usages.

Caution: Reducing the allocation of workspace will reduce the memory requirements of the processor and may result in execution speed increases on some machines. However, such reductions increase the likelihood of a processor abort. The user is forewarned that actual utilization is *very* sensitive to slight modeling changes, to differences in cases run, and to the solution routine employed.

6.9.6 FLUINT Simulation Routines

This section describes the following simulation routines:

COMPRS .... Compressors and other turbomachinery
BELACC .... Two–phase Bellows Accumulators
NCGBUB .... Stationary Noncondensible Gas Bubble
MIXGAS .... Mixing of Perfect Gases
NONEQ0/1 .. Nonequilibrium two-phase control volumes
CHKCHP/L .. Choked flow (sonic li-) detection and simulation aids
CHOKER .... Facilitated choked flo.. (sonic limit) simulation aid f.. connectors
COMPLQ .... Add liquid compressibility to tanks; water hammer simulation aid
WAVLIM .... Maximum time step for capturing water hammer and acoustic waves

Most of these routines perform adjunct solutions, helping to calculate FLUINT network parameters such as COMP, VDOT, GK, HK, etc. for tanks, NULL connectors, etc. input by the user.

6.9.6.1 Compressors and Other Turbomachinery

Alls paths are inherently adiabatic and therefore isenthalpic: the energy input associated with pumping is neglected. While this is normally acceptable for viscous flow losses and for pumps, fans, and circulators, it is nonsensical for compressors and turbines. Thus, a simulation of such turbomachinery requires the attributes of both paths and lumps.

Subroutine Name: COMPRS

Description: This routine allows the user to simulate generalized compressors. The required inputs include a total input power and an isentropic efficiency η. The same routine may be used to simulate turbines, although its usage in that context is more limited since the usual analytic approach is to calculate the output power, rather than to define it as an input.

To use this routine:

1) Build a NULL connector to act as the compressor. (A fake initial value of GK must be supplied to satisfy the input requirements for a NULL connector.)

2) *The QDOT on the defined downstream lump is the power input that drives the compressor*; it may be changed as needed.

3a) Call COMPRS in FLOGIC 0. When called from this location, the resulting NULL flowrate driving factor HK may be damped. The QDOT for the downstream lump will not yet be updated if it is tied to any nodes.*

3b) Or, call COMPRS in FLOGIC 1. When called from this location, the resulting NULL flowrate driving factor HK will be undamped. The QDOT for the downstream lump will have been updated if it is tied to any nodes.*

Using a tank as the downstream lump can cause the flowrates to differ transiently from the equilibrium value since the tank cannot react instantly to the actions of the compressor. Internal damping to prevent flowrate jumps or oscillations (if called from FLOGIC 0) and use of a junction may cause temperature excursion warnings since the flowrate may be occasionally prevented from rising or falling to its desired level instantly.

Calling Sequence:

```
CALL COMPRS ( FSMN, NULL, EFF)
```

where:

FSMN ..... fluid submodel name, in single quotes
NULL ..... the ID of the NULL connector representing compressor
EFF ..... Isentropic efficiency η; less than 1.0 for a compressor, or greater than one ($1/\eta$) for a turbine.

---

* Ties to any lump override the user-imposed QDOT. For tanks in solution routines other than FASTIC, the QDOT of a tied tank may be altered in FLOGIC 1. Although not illegal, there is currently no known reason why a user might wish to use the QDOT of a tied lump. A "fixed" or non-heat transfer based QDOT is the expected usage.

Zero QDOT causes the NULL to act like a shut valve to be consistent with the governing equations. Negative QDOT is equivalent to a turbine. In this case, EFF should be greater than or equal to 1.0: it should be set to the inverse of the turbine η.

Restrictions: COMPRS should not be used with liquids or two-phase flow. It can be used with library fluids, and 7000 and 8000 series user fluids. To use with 6000 fluids, the entropy property routine VS must be defined in the FPROP block. If two–phase fl · appears at the inlet, a warning will be produced and the simulation will attempt to continue. However, if the heat input is insufficient to result in vapor at the outlet, property routine errors and/or an abort may result.

6.9.6.2 BELACC: Bellows Accumulator Simulation

The following routine is intended to simulate an accumulator in a two–phase system where liquid and vapor from the same loop are plumbed into opposite sides of a bellows. However, the routine is very general, and has been used to model the spring–like behavior of a meniscus and/or deformable wick located between the liquid and vapor sides of a capillary evaporator pump. It represents a unique capability to define a pressure relationship between two lumps without the need for mass exchange (i.e., a path).

Subroutine Name: BELACC

Description: Simulates a bellows accumulator, or any control volume that is subdivided into two tanks separated by a membrane or spring–like divider. The divider obeys a linear spring relationship: $PL_{liq} - PL_{vap} = DPDV*(VOL_{liq} - VZERO)$, where DPDV is the volumetric spring rate and VZERO is the resting "position" (perhaps negative).

To use this routine:

1) Build two tanks in one submodel. One of these tanks should contain liquid and the other vapor, although the routine handles all conditions except lack of vapor in the designated vapor side.

2) Before calling STDSTL or transient routines, make sure that FASTIC has been called or that the initial conditions are correct (i.e., that the volumes sum to the correct value, and that the pressure differential corresponds to the initial volumes).

3) Call BELACC in FLOGIC 1, according to the formats specified below.

The tanks are assumed to be on opposite sides of the bellows. There are *no* restrictions on the number or type of paths or ties that may be applied to these tanks, including paths between the tanks (such as might be used to represent leaks in the bellows), or ties to a common node (which may be used to represent heat transfer through the bellows wall).

In FASTIC, one or both of the tanks may be held by HLDLMP. At the completion of FASTIC, the volumes of the tanks will have been adjusted according to the pressure differential as needed to provide consistent initial conditions for later solution routines.

Restrictions: Must be called in FLOGIC 1. At least some vapor must exist in the designated vapor tank; the routine will abort if this tank becomes hard filled with liquid. The total volume must be greater than the maximum liquid volume, which must be greater than the minimum liquid volume, which must be greater than zero: VTOT > VMAXL > VMINL > 0. The bellows volumetric spring force may not be negative, although zero is specifically permitted. The user should not manipulate tank parameters (especially VDOT, COMP, and QDOT) *after* the call to BELACC.

Calling Sequence:

```
      CALL BELACC(MODEL, NLIQ, NVAP, VTOT, VMAXL,
     + VMINL, VZERO, DPDV)
```

where:

MODEL .... Fluid submodel name containing the tanks, in single quotes

NLIQ ..... User ID of the liquid side tank

NVAP ..... User ID of the vapor (or two–phase) side tank

VTOT ..... Total volume of the accumulator

VMAXL .... Maximum volume that the liquid tank can occupy; VMAXL < VTOT.

VMINL .... Minimum volume that the liquid tank can occupy; 0 < VMINL < VMAXL.

VZERO .... Liquid volume at which the bellows force is zero (i.e., the position at which the liquid tank pressure equals the vapor tank pressure). May be any number, including negative or greater than VTOT. Analogous to x_o in the spring force equation: $F = k(x-x_o)$.

DPDV ..... Bellows volumetric spring rate (assumed constant over the interval), in units of pressure over volume. Must be nonnegative; zero is legal. Analogous to k in the spring force equation: $F = k(x-x_o)$.

Example:

```
      CALL BELACC('TONY', 10, 20, 1.0, 0.95, 0.05,
     + 0.5, 2./0.45)
```

In the above example, assuming English units, the accumulator volume is 1.0 ft^3, and the liquid side is prevented mechanically from becoming larger than 0.95 ft^3 or small than 0.05 ft^3. Before filling, the bellows is at rest in the middle of its swing; the pressures differential is zero at this position. Because DPDV=2.0/0.45, the liquid side (tank TONY.10) will be at least 2.0 psi higher than the vapor side when the accumulator is full, and at least 2.0 psi less than the vapor side when empty. (The phrase "at least" is necessary in the last sentence since, as an example, the pressure in the liquid will continue to rise above 2.0 psid if it has reached VMAXL and fluid is still attempting to enter the liquid side or exit the vapor side.) If VZERO had been input as 0.05, the resting position would be the same as the empty state.

For nonlinear spring modeling, DPDV and VZERO may be varied, but changes should be very gradual—preferably dependent on time or volume, and the new spring force should be nearly continuous with the previous equation at the current bellows position (liquid tank volume).

6.9.6.3 NCGBUB: Noncondensible Gas Bubble Simulation

The following routine is intended to simulate a noncondensible gas (NCG) bubble trapped within a subcooled liquid control volume. The gas is assumed to be a perfect gas in thermal equilibrium with the liquid in the control volume, and is assumed to not dissolve in it. The gas bubble may grow or shrink according to its partial pressure, displacing liquid in the process. The gas itself is not modeled by FLUINT, merely the effects of the gas on the rest of the control volume, which is modeled by a FLUINT tank.

As the subcooling diminishes, the gas volume will grow and the liquid volume will shrink. The minimum gas volume is 0.1% of the total. Dissolving of gas and subsequent gas evolution is neglected, as is the mass and energy associated with the working fluid vapor in the void and the related evaporation and condensation. The gas bubble is assumed to be trapped within the control volume by gravity, swirling, or wicks.

Subroutine Name: NCGBUB

Description: Simulates a trapped (stationary) NCG bubble within a subcooled control volume, whose liquid portion is modeled by a tank.

To use this routine:

1) Build one tank to represent the liquid in the control volume.
2) Before calling NCGBUB or transient routines, make sure that FASTIC has been called or that the initial conditions are correct (i.e., that the liquid is subcooled, and that the volume corresponds to the total minus that which would be occupied by gas).
3) Call NCGBUB in FLOGIC 1, according to the formats specified below.

In FASTIC, the tank may be held by HLDLMP. At the completion of FASTIC, the volume of the tank will have been adjusted according to its temperature and pressure as needed to provide consistent initial conditions for later solution routines.

Restrictions: Must be called in FLOGIC 1. Tank must be subcooled. Zero gas mass is not allowed. If no gas is present, set VDOT=COMP=0. and do not call NCGBUB. Otherwise, the user should not manipulate tank parameters (especially VDOT, COMP, and QDOT) *after* the call to NCGBUB.

Calling Sequence:

```
CALL NCGBUB(MODEL, NLIQ, EMARR, VTOT)
```

where:

MODEL .... Fluid submodel name containing the tank, in single quotes
NLIQ ..... User ID of the liquid tank
EMARR .... Amount of gas present, expressed in terms of the mass times the gas constant: mR from the perfect gas relation PV=mRT.
VTOT ..... Total volume of the control volume (i.e., the liquid tank volume plus the fictitious gas void volume).

In metric units, EMARR has units of J/K. The universal gas constant is 8314 J/kmol–K. EMARR is therefore the mass of NCG in kg, times the universal constant, divided by the molecular weight (kg/kmol). Or, input EMARR as the number of kmol times the universal constant.

In English units, EMARR has units of psia–ft^3/R. The universal gas constant is 1545 $ft–lb_f$/lbmol–R. EMARR is therefore the mass of NCG in lb_m, divided by 144.0, times the universal constant, divided by the molecular weight (lb_m/lbmol). Or, input EMARR as the number of lbmol times the universal constant, divided by 144.0.

Example:

```
CALL NCGBUB('RUBY', 100, 1.0E-12*8314.,1000.E-6)
```

In the above example, assuming metric units, one liter of control volume contains 1.0E–12 kmol of noncondensible gas. Remember the volume of tank 10 will always be less than one liter. To initialize the tank volume before calling NCGBUB or FASTIC, the following lines of logic might apply:

```
 TL10   = TL10 - ABSZRO
 VTEST  = VTOT - EMARR*TL10
+       /(PL10-VPS(TL10,RUBY.FI)-PATMOS)
 TL10   = TL10 + ABSZRO
 CALL CHGVOL('RUBY',10,VTEST)
```

To model gas generation, transport, evolution, dissolving, etc., EMARR may be varied, but changes should be very gradual—preferably dependent on time, temperature, or volume.

6.9.6.4 MIXGAS: Simulation of Perfect Gas Mixing

The following routine simulates a control volume shared by two perfect gases, an assumption that enables the use of the partial volume approach. Within the control volume, the FLUINT tank volume of each species is equal to the mole fraction of that species times the total real volume. Upstream lumps will "feel" the hydrodynamic effects of both gases, and the thermodynamics within the control volume (compression heating, mixing, etc.) will be correctly predicted.

Subroutine Name: MIXGAS

Description: Simulates a control volume consisting of two partial volumes, each of which is modeled by a gas– or vapor–filled FLUINT tank that are usually built in different submodels.

To use this routine:

1) Build two tanks in two submodels, each of which contain a gas or vapor that behaves perfectly within these tanks, if not everywhere within the submodels.

2) Before calling STDSTL or transient routines, make sure the initial conditions are correct (i.e., that the volume sum to the correct value, that the pressures and temperatures are equal, and that the volumes correspond to the mole fraction ratio). FASTIC cannot perform these initializations, and so MIXGAS does nothing when called from that routine.

3) Call MIXGAS in FLOGIC 1, according to the formats specified below.

Restrictions: Must be called in FLOGIC 1. Both tanks must contain only gas (XL=1.0). The user should not manipulate tank parameters (especially VDOT, COMP, and QDOT) *after* the call to MIXGAS. Should only be called once per set of tanks—it should only appear in FLOGIC 1 for one of the submodels involved.

The fluids need not be 8000 series fluids as long as they both behave perfectly within the tanks, obeying $P/(\rho T)$ = constant, where P and T are in absolute units. Most real descriptions behave perfectly in the limits of high temperature and low pressure. Less than perfect behavior is not catastrophic; it will be evidenced by a non–negligible pressure difference between the two tanks.

Initializing the two tanks represents some problems since FASTIC cannot perform such initializations, as it does for similar simulation routines. The user must start the problem with the temperatures and pressures equal. If the initial mole fractions, partial pressures (proportional to both mole fractions and partial volumes), or masses are known, then the two volumes can be computed easily knowing the total volume. Otherwise, *users cannot set the initial volumes arbitrarily* even if the pressure and temperatures are equal and the volumes sum to the total. Given two species, a temperature, and a pressure, there is only solution; only one pair of tank volumes will satisfy the partial volume relationship. If precise initial conditions are not known, the user is best off assuming a certain initial mole fraction for each species (the sum of both should be unity), and then setting the volumes accordingly.

Calling Sequence:

```
CALL MIXGAS(MODL1, NTNK1, MODL2, NTNK2, VTOT)
```

where:

MODL1	Fluid submodel name containing tank NTNK1, in single quotes
NTNK1	User ID of the first tank (in submodel MODL1)
MODL2	Fluid submodel name containing tank NTNK2, in single quotes
NTNK2	User ID of the second tank (in submodel MODL2)
VTOT	Total volume of the whole control volume

Example:

```
CALL MIXGAS('GAS', 1000, 'GUS', 1000, 30.0)
```

In the above example, assuming English units, the control volume is 30.0 ft^3. During simulation, the pressures and temperatures of GAS.1000 and GUS.1000 will be approximately equal, and the volumes will reflect the relative mole fractions.

The two tanks may exist within the same submodel, although there appears to be no reason for such an approach. Perhaps a more useful extension of MIXGAS is to allow division of one submodel into two, exploiting the speed improvements that can result from analyzing two smaller submodels instead of one large submodel. In other words, the working fluid can be the same in both tanks. Perhaps future versions of BELACC and NONEQ1 will also support such usage, allowing the analyst to break a two–phase submodel into two single–phase submodels.

Finally, the user should remember that the two gases are not *really* mixed, and that outflow from either of these two tanks will consist of the same substance used in the rest of the submodel. The user might consider using a third submodel to model outflow, although the assumption of a fixed composition is implied. Such an assumption would be valid only if the filling of the upstream vessel is complete, or if the inflow rates tend to keep the composition constant.

6.9.6.5 Nonequilibrium Two-Phase Control Volumes

Introduction: The Need for Nonequilibrium Simulations

A fundamental FLUINT assumption is that any lump is characterized by a single equilibrium thermodynamic state. In fact, that is the *definition* of a lump, just as a node is defined as a point with a single temperature. The assumption is equivalent to one of perfect mixing within each lump, meaning that the heat transfer and mixing processes are assumed to occur over a very small time scale compared to the system solution. While that assumption is normally justified and greatly enhances solution speed, it can be completely inappropriate in some instances.

Within a FLUINT tank containing both liquid and vapor (e.g., $0 < XL < 1$), the perfect mixing assumption means that any changes or additions to either phase *immediately* affects the lump pressure—the heat and mass transfer rates between phases are assumed to be infinite. In Sample Problem C (the pressure cooker), this assumption causes the pressure in the cooker to drop exponentially with time because even a little subcooled liquid drastically reduces the equilibrium saturation pressure. In many cases, such perturbations are unstable. Slight injection of liquid can lead to catastrophic collapse of the vapor since the liquid influx will increase as the control volume pressure drops. In a duct model built with a LINE or HX macro and tanks, internally applied spatial acceleration terms usually prevent such behavior even though minor oscillations occur: sharp flowrate gradients are "discouraged" since they cause self-correcting pressure gradients.

One notable example in which modeling of nonequilibrium behavior is crucial to the overall system response are two-phase reservoirs, which are commonly proposed in both capillary- and mechanically-pumped spacecraft two-phase coolant loops. These devices accommodate changes in liquid inventory and maintain set-point conditions by controlling the temperature of a fluid volume. Such reservoirs are heated or are injected with warm vapor, and are cooled with a combination of a cold-biased environment, conduction to subcooled piping, and occasional or purposeful injections of cold liquid. Suppose such devices were modeled with plena or tanks. If a single two-phase tank were used, then the reservoir becomes artificially susceptible to "cold shock," which is the aforementioned catastrophic collapse of the vapor bubble. In reality, with less than infinite heat and mass transfer, injection of cold liquid may actually cause a temporary pressure *rise* due to compression of the vapor bubble. In fact, the vapor bubble can be very difficult to collapse completely because of this effect. At the very least, small time steps could result because of this inherently "delicate" situation.

Of course, there would be no point discussing this topic in such detail if an alternative were not available. Read on.

Approach to Nonequilibrium Simulation

Finite heat and mass transfer rates can be accommodated within two-phase FLUINT control volumes by:

1) Using *two* tanks instead of one to model the control volume, and adding a NULL connector between the two tanks. (A fake initial value of GK must be supplied to satisfy the input requirements for a NULL connector. Also, as will be discussed later, the NULL connector may optionally be twinned.)

2) Calling routines NONEQ0 and NONEQ1 in FLOGIC 0 and FLOGIC 1, respectively, for each pair of nonequilibrium tanks.

One tank is used to model the liquid phase, and the other the vapor phase. The NULL connector is used to model the mass transfer between the phases. As follows from the above discussion, the fluid within each tank is assumed perfectly mixed: the liquid tank represents the bulk liquid state, and the vapor tank represents the bulk vapor state. There are no additional limits on the number or type of paths or ties attached to these tanks.

The COMP and VDOT values of both tanks and the flowrate and resistance values (HK and GK) of the NULL connector are all manipulated by the simulation subroutines (NONEQ0 and NONEQ1) to maintain the following relationships:

$$P_{liq} + \Delta P_{capillary} = P_{vap}$$

$$V_{liq} + V_{vap} = V_{tot}$$

where:

P_{liq} ..... Liquid tank pressure
$\Delta P_{capillary}$ .... Capillary pressure term (zero to $2\sigma/r_c$, where r_c is the 2D capillary radius)
P_{vap} ..... Vapor tank pressure
V_{liq} ..... Liquid tank volume
V_{vap} ..... Vapor tank volume
V_{tot} ..... Total volume for control volume

Unfortunately, these relationships represent extra constraints that cannot be accommodated exactly. In other words, the COMP, VDOT, GK, and HK values are *estimated* at the start of the time step, but may not be entirely appropriate because of unforeseen changes in the system solution. Such is the case for nearly all FLUINT user-manipulated values. Fortunately, the simulation subroutines keep track of error terms (e.g., the actual total volume minus the desired total) and automatically correct for them. The net effect is that the above equations are adequately satisfied for most analytic requirements. Exceptions to this statement are noted below.

In addition to the above constraint equations, the heat and mass transfer equation is as follows:

$$U_{vap} \cdot A_{surf} \cdot (T_{vap} - T_{sat}) - U_{liq} \cdot A_{surf} \cdot (T_{sat} - T_{liq}) = m \cdot \Delta H(T_{sat})$$

where:

U_{vap} ..... Bulk vapor to interface heat transfer (mixing) coefficient
A_{surf} ..... Interface surface area
T_{vap} ..... Vapor tank temperature
T_{sat} ..... Interface (saturation) temperature
U_{liq} ..... Bulk liquid to interface heat transfer (mixing) coefficient
T_{liq} ..... Liquid tank temperature
m ..... Surface vaporization or condensation mass rate
$\Delta H(T_{sat})$ ..... Delta enthalpy at interface (approx. equal to the heat of vaporization)

The heat transfer coefficients and interface area are user-supplied, although defaults are available.

Ideally, the quality in the liquid tank is always 0.0, and the quality in the vapor tank is always 1.0. In reality, however, there are several cases where this is not quite true.

First, bulk boiling in the liquid control volume and/or bulk condensation (fogging) in the vapor control volume may exist. Under these circumstances, mass is moved from one tank to the other to attempt to maintain separation of phases. For reasons of computational accuracy and time step control, it is not desirable to completely separate the phases. Rather, upper and lower limits are set on the amount of vapor tolerated in the liquid volume and vice versa. During bulk boiling for example, the quality of the liquid tank hovers around a low value (about 3.E-5 by default, or as specified via the NONEQS routine) while the vapor volume grows and the liquid volume shrinks. When boil–off and/or condensation are continuous, some computational advantages can be exploited by twinning the NULL connector. This action automatically invokes special logic to transport the excess phase (e.g., vapor during boil–off) in the "spare" twin, which has the effect of stabilizing the quality in the upstream tank (e.g., the liquid tank during boil–off)

Second, the fluid in the control volume may become entirely vapor or entirely liquid. In that instance, the volumes gradually become equal, the NULL connector is opened (e.g., the resistance is lowered to equalize the pressures and to allow flow through the volume), and mixing is promoted between the two sides. *This is largely the same logic applied in FASTIC.*

The transition between single-phase and two-phase modes of operation deserves some discussion. Because it is not desirable to track a diminishingly small control volume, the shrinking is halted when the void fraction in the entire control volume is less than 0.01 or greater than 0.99 (by default, or as specified via the NONEQS routine). At that point, mixing is promoted between the two tanks, and the NULL connector is opened. Mixing continues gradually each time step until the volumes become equal, at which time they exchange up to 1% (by default) of their fluid each time step. This exchange does not involve flow work: it can be though of as a simple "redrawing" of the control volume boundaries, and is not intended to simulate any real mixing or heat transfer process.

Subroutine Name: NONEQ0

Description: NONEQ0, which *must* be called from FLOGIC 0, performs two functions. First, it decides whether or not to use separated-phase logic or single-phase (mixed) logic during the next solution step. If within FASTIC or if the separated method would result in one tank becoming unacceptably small, then single-phase logic is used. Second, based on the first decision phases are either mixed or separated (if need be) into the appropriate control volume.

Calling Sequence:

```
CALL NONEQ0 (FSMN, LLIQ, LVAP, NULL)
```

where:

FSMN	fluid submodel name, in single quotes
LLIQ	ID of the liquid tank
LVAP	ID of the vapor tank
NULL	the ID of the NULL connector between LLIQ and LVAP, or the ID of the primary twin if that connector has been twinned

Example:

```
CALL NONEQ0('CPL', 1000, 2000, 1002)
```

Subroutine Name: NONEQ1

Description: NONEQ1, which *must* be called from FLOGIC 1, performs either of two functions. If the previous NONEQ0 call determined that single-phase (mixed) logic is required, then an appropriate resistance is calculated for the NULL connector, and it is opened to allow flow between the tanks. The resistance is chosen to be low compared to local or submodel resistances. The COMP and VDOT values for both tanks are set to zero, and the QDOTs are not affected.

Alternatively, if separated-phase logic is required, then NONEQ1 calculates the rate of mass and heat transfer between the two tanks, and sets the flowrate of the NULL connector and adjusts the QDOTs of the liquid and vapor tank to simulate these effects. If continuous phase change (e.g., boil–off or fogging) is being experienced and the NULL has been twinned, the flowrate in one the the twins will be set to the rate of phase change to help smooth the integration. NONEQ1 then calculates values of COMP and VDOT for the liquid tank, and VDOT for the vapor tank (COMP for this tank is always zero) such that the pressure and volume constraint equations are met. NONEQ1 also sets time step limits to control changes to the differential pressure between the two tanks, using the same methods as do CAPIL connectors and CAPPMP macros. A time step message of 'CAPILLARY CHANGE LIMIT' will be produced if this limit caused the previous time step.

Calling Sequence:

```
 CALL NONEQ1 (FSMN, LLIQ, LVAP, NULL, ULIQ,
+             UVAP, ASUR,  RCW, VTOT)
```

where:

FSMN ..... fluid submodel name, in single quotes

LLIQ ..... ID of the liquid tank

LVAP ..... ID of the vapor tank

NULL ..... the ID of the NULL connector between LLIQ and LVAP, or the ID of the primary twin if that connector has been twinned

ASUR ..... the interfacial surface area. If negative, then the value used is |ASUR| times the surface area of the vapor tank assuming a sphere.

ULIQ ..... the liquid to interface heat transfer coefficient. If negative, then the value used is |ULIQ| times a value based on solid conduction: k*ASUR/L, where k is the conductivity of the liquid and L is the characteristic length of the liquid conduction path. L is calculated as two times the volume of the liquid tank divided by the interface surface area ASUR.

UVAP ..... the vapor to interface heat transfer coefficient. If negative, then the value used is |UVAP| times a value based on solid conduction: k*ASUR/L, where k is the conductivity of the vapor and L is the characteristic length of the vapor conduction path. L is calculated as two times the volume of the vapor tank divided by the interface surface area ASUR.

RCW ..... the radius of curvature at the interface (normally huge unless some form of capillary structure is present). Used to calculate the pressure difference between the phases. This term is zero if RCW is 1.0E20 or greater. Based on a spherical vapor bubble if RCW is 0.0 or less.

VTOT ..... total volume of control volume (hint: tolerancing, as is shown in the example below, has been found to increase solution speed with negligible impact on accuracy)

Example:

```
 VTEST    = 0.1*VOLRES + 0.9*(VOL1000 + VOL2000)
 CALL NONEQ1 ('CPL', 1000, 2000, 1002, -1.0, -10.0,
+             -1.0, 1.0E30, VTEST)
```

Restrictions: Must be called in FLOGIC 1. The user should not manipulate tank parameters VDOT, COMP, or QDOT for either tank *at any time*. Changes to attached ties and paths are legal *before* the call to NONEQ1, as are changes to any input parameter such as UVAP or VTOT. If VTOT is changed, the changes should be very gradual, preferably a function of time or some time–dependent parameter such as pressure.

Subroutine Name: NONEQS

Description: NONEQS is the optional set-u ıe that can be used to customize the NONEQ solution by altering various default parame: ınd tolerances (e.g., the minimum volume fraction allowed for nonequilibrium simulations). Five values may be specified, as described below. *Any updates to these values will affect all nonequibrium solutions in all submodels.*

NONEQS may be called as many times as needed to set all desired parameters, and should be called from within OPERATIONS DATA before ·ating any solutions. Changing values at other times can have unforeseen effects, especially ·nquilibrium solutions are employed in more than one submodel. If inputs are inconsistent w.ıh other values or exceed the valid range, a warning will be produced and the limiting value will be used.

The names of the five values, their meaning, and gui.elines and restrictions on their values are described below.

VLIM – Lowest allowable volume fraction for nonequilibrium solutions. If either the liquid tank or the vapor tank becomes less than VLIM (by default 0.01 or 1%), mixing will begin and the nonequilibrium logic will be suspended. Similarly, nonequilibrium solutions will not start until the void fraction is greater than VLIM or the liquid fraction is greater than VLIM. VLIM cannot be less than 1.0E–4 nor greater than 0.4. *The user is cautioned that a value too small can result in unstable integrations and diminishingly small time steps, and that even the default value has been f·· nd to be too large in some models.*

XHIV – Up nd of hysteresis band for vapor t ·k quality. If the vapor tank quality drops too low (see XI ɔelow), it will be instantaneously reset to XHIV in NONEQ0 and the corresponding amount of liquid removed from the vapor tank will be added to the liquid tank. The default value is 0.995. XHIV cannot be greater than 0.9999 nor less than 0.1. It cannot be less than or equal to XLOV.

XLOV – Lower end of hysteresis band for vapor tank quality. If the vapor tank quality drops below XLOV, it will be instantaneously reset to XHIV in NONEQ0 and the corresponding amount of liquid removed from the vapor tank will be added to the liquid tank. The default value is 0.99. XHIV cannot be greater than or equal to XHIV, nor less than 0.1. It cannot be less twice the value of XHIL (see below).

XHIL – Upper end of hysteresis band for liquid tank quality. If the liquid tank quality rises above XHIL, it will be instantaneously reset to XLOL (see below) in NONEQ0 and the corresponding amount of vapor removed from the liquid tank will be added to the vapor tank. The default value is 1.0E–4. XHIV cannot be less than nor less than 1.0E–5 nor less than twice XLOL. It cannot be greater than half of XLOV.

XHIL – Lowe· of hysteresis band for liquid tank quality. If the liquid tank quality rises above XHIL, it v· ·· instantaneously reset to XLOI· in NONT ·) and the corresponding amount of vapor r· ·d from the liquid tank will be ad to the v tank. The default value is 1.0E–5. XHIV · .ot be less than nor less than 1.0· ıor gre· than half XHIL.

Calling Sequence:

```
CALL NONEQS (VNAME, VALUE)
```

where:

VNAME .... name of value to change, in single quotes. Must be one of the following: 'VLIM', 'XHIH', 'XLOV', 'XHIL', or 'XLOL'

VALUE .... the real (floating point) value to use for the parameter VNAME

Example:

```
CALL NONEQS('VLIM', 0.02)
```

Caution: NONEQS should be called from within OPERATIONS DATA before initiating any solutions. Changing values at other times can have unforeseen effects, especially if nonquilibrium solutions are employed in more than one submodel. Otherwise, if changes must be made during a solution, they must be made in FLOGIC0 before any call to NONEQ0.

Caution: Instabilities and small time steps can result if (1) VLIM is too small (even the default value is too small for some models!), (2) XHIV is too large, (3) XLOL is too small, (4) the difference between XHIV and XLOV is too small, or (5) the difference between XHIV and XLOV is too small. There are no firm guidelines with respect to which values cause problems.

Behavior and Usage

The chosen methods were purposely generalized to allow these special control volumes to be used in detailed simulations of a variety of specific hardware. There are no limits placed on the number or type of ties or paths to either tank, allowing fairly complex situations to be modeled. In reality, however, the user will find some combinations unacceptable for either modeling reasons or solution efficiency. For example, a CAPPMP macro or CAPIL connector may be placed between the two tanks. However, if it deprimes cold liquid can enter the warm vapor side, causing it to collapse, or warm vapor can enter the cold liquid side, causing its pressure to surge as it warms rapidly. Moreover, an artificial deprime mechanism exists because the liquid pressure may occasionally rise slightly higher than the vapor pressure due to error terms (especially if no capillary term is added). The user should not be discouraged from attempting such complex modelling, but should be aware of the potential interactions.

There is no permanent significance between the designated liquid vs. vapor tank: users can switch these designations during the run as long as they are consistent between NONEQ0 and NONEQ1 calls (e.g., make the switch before NONEQ0 and stick to it in the subsequent NONEQ1 call). Although the author cannot presently conceive of a use for this fact, someone no doubt will. Of course, the sudden shifting of liquid from one tank to another may have indirect effects on adjacent ties and paths.

Default Values and Modeling Choices—Perhaps the biggest obstacle facing the user is the choice of the values ASUR, ULIQ, and UVAP. These are dependent on the geometry of the control volume, its orientation relative to body forces, the movement of the fluid in each phase relative to the other phase (which is influenced by ingress and egress if any). These values can rarely be determined analytically: the user must rely on parametric analysis, judgment, and calibration with test data if available. The following discussion is intended to help establish good engineering judgment, and to explain the usage of the default options.

The defaults for ASUR, ULIQ, and UVAP are intended to be used extensively, and can be modified without necessarily completely overriding them. To use the default values, simply input -1.0. To use twice the default value: -2.0, half the default value: -0.5, etc. Otherwise, non-negative values will be used directly. The intent is to allow the user to parameterize the problem in terms of ratios of the default values if no other information is available. While the limit of zero heat and mass transfer can be handled by setting both ULIQ and UVAP to zero, the limit of infinite heat transfer should be handled by model the control volume with a single tank.

The parameter with the least uncertainty is ASUR, the interfacial surface area. The default is based on a spherical vapor bubble, which is valid in the limit of small void fractions for *any* shaped control volume as long as all the vapor is assumed to exist in one bubble. Otherwise, the value of ASUR may be estimated based on the shape of the control volume, any imposed body forces, and the fill level. For example, assume a cylindrical control volume of length L and diameter D. For a vertical cylinder in a gravity field, the surface area is a disk of diameter D. In the absence of gravity or if D is very small, the surface area is a hemisphere of diameter D (up to a certain fill level). In tilted cylinder, the surface area is either an elliptical disk (in gravity) or an ellipsoid (low gravity or small D). The user should not be overly concerned with calculating an exact surface area because this term is always used in combination with ULIQ and UVAP, and the uncertainty in those parameters is relatively large, as will be discussed below.

The defaults for ULIQ and UVAP are based on the input or calculated value of ASUR, and are estimated assuming solid conduction: KA/L, where K is the conductivity of the appropriate phase, A is ASUR (whether input or defaulted), and L is a characteristic conduction path length based on the current volume of the appropriate phase and ASUR: L = VOL/(2*ASUR). The validity of this approximation varies with control volume and surface area shape, but it provides a reasonable estimate of the lowest possible heat transfer coefficient in all cases. Surprisingly, *this lowest estimate of solid conduction also often the best estimate*. The reasons for this statement are as follows.

First, there is usually little boundary layer shear at the interface: the vapor is often just dragged along with the liquid surface motion, if any. In one experimental study, a cup containing vapor was inverted in a spinning pool of subcooled liquid. The heat transfer rate in the vapor was very low despite the relative motion of the liquid; apparently solid body rotation formed quickly in the vapor. The moral is that, unlike heat transfer between a fluid and a wall, the velocity profile at a liquid to vapor interface does not contain much of the shear layer that normally increases heat transfer beyond pure conduction.

Second, stratification or poor mixing in the liquid volume is common if there is only inlet or outlet to that volume (as in the case of a CPL—capillary pumped loop—reservoir), if the control volume is vertical and has a long aspect ratio (e.g., a vertical or slightly tilted cylinder), or if some sort of baffle or disrupter exists at the inlet to disperse incoming fluid and prevent jets from forming. Even without the above conditions, a solid conduction value may be warranted if

the liquid is not disturbed and if the flow *exits* from the liquid control volume: such conditions do not promote mixing of the liquid. Conditions that *do* promote mixing and hence warrant a higher ULIQ include ingress of undistrupted liquid into a disturbed or short aspect ratio control volume, and cases where liquid flows through the control volume. In the former case the formation of liquid jets is promoted: incoming cold liquid is able to reach the interface easily in the form of a jet, and the resulting rapid condensation at the interface drops the control volume pressure, increasing the jet flowrate.

In summary, with no vapor ingress, the default conduction value should be used for UVAP, and only minimal increases should be considered even if there is flow into or through the vapor portion. For the liquid coefficient ULIQ, the choices are less clear, although a solid conduction value is appropriate for most cases in which mixing in the liquid is not promoted. As a rule of thumb, relying on the facts that ASUR and UVAP can be reasonably estimated, the user should concentrate on ULIQ for calibration against test and for parameterizing system response, preferably as a ratio of the solid conduction default (see the example below). Remember that this ULIQ ratio may change during execution as conditions (e.g., the inflow rate) change.

Noting that the pressure is the control volume is governed by the saturation condition of the vapor volume, *the user should not neglect or oversimplify heat transfer to the container wall nor neglect the mass of the wall in transient analyses where the pressure response of the control volume is critical.* These wall terms may include natural convection and film condensation in the vapor space, and may include subcooled and nucleate or pool boiling in the liquid space. Otherwise, solid conduction or single-phase convection terms should be considered as a minimum for wall heat transfer. The PUTTIE routine is useful with such detailed wall models, as illustrated in Sample Problem F.

Two-phase (separated-phase) Behavior—To illustrate typical behavior of a nonequilibrium two-phase control volume, take the case of a CPL reservoir. Such a reservoir usually takes the form of a vertical cylinder connected to a subcooled liquid return line by a tube plumbed at the bottom. A thermostatically controlled resistance heater on the wall maintains the loop set point. Now consider cold liquid flowing into the reservoir. If the influx is more rapid than the condensation rate at the interface, the vapor will compress and heat up, driving the saturation pressure up temporarily and tending to discourage the influx. Eventually the vapor will cool down as the influx stops, and will begin to condense (not just at the interface), lowering the pressure and thus permitting more influx. If the influx were more gradual, the control volume may never experience an increase in pressure. Note that equilibrium conditions dictate that the pressure will eventually be less than before unless the wall heater can keep up with the rate of cold liquid influx.

In the reverse process, if liquid leaves gradually (less than the evaporation rate at the interface), then the pressure will only drop slightly if at all. However, if the liquid is quickly drawn out of the reservoir, the pressure will drop quickly as the vapor is expanded, perhaps to the point where the bulk of the liquid begins to boil. At that point, further pressure reductions of any significance do not occur: the vapor volume grows from the coalescence of bubbles from the liquid side.

In the first instance, the rate of condensation at the interface is *almost strictly a function of the heat transfer and mixing in the liquid side: ULIQ.* Thus, the distinction between "rapid" vs. "slow" influx is dependent on the value of ULIQ. This fact, combined with the facts that the heat transfer in a vapor bubble is almost entirely due to conduction and that the pressure in the con-

trol volume is a function of the vapor side, causes analytic difficulties in the limits of a nearly hard-filled reservoir. The vapor bubble is persistent. Increasing the liquid influx slightly causes the vapor to superheat quickly, and yet the state of this small bubble dominates the pressure of the larger liquid volume. At some point in the analysis, the decision to abandon nonequilibrium methods and to revert to single-phase methods must be made, as will be described next.

Single-phase (mixed) Behavior and Transitions—In the single-phase mode, "mixing" or the redrawing of control volume boundaries is performed such that the individual volumes gradually become equal in size. Also, the NULL connector is given a small resistance to promote mass exchange and pressure equalization, although the pressure difference cannot always be exactly zero (which would require an illegal zero resistance). This is the only simulation possible in FASTIC, which is by nature instantaneous and at equilibrium. If unequal sized tanks are desired for whatever reasons, the user can call CHGVOL before or after NONEQ0 to keep the tanks unequal. Mass and energy will not be completely conserved, but the error should be small as long as both tanks are in nearly the same state, as is the case except immediately after transition from separated-phase logic.

As mentioned above, transitions must be made between single-phase and separated-phase methods in the limits of small tank volume. These transitions occur when the total void fraction is 0.01 or 0.99 by default (these limits can be customized if needed using the NONEQS set–up routine). In other words, at no time is any tank allowed to contain less than 1% of the total volume. Unless high boil–off rates are experienced, the transition to and from single-phase vapor is usually smooth. Because of the "persistent bubble" problem mentioned in the previous section, the same cannot be said for the transitions to and from single-phase liquid. When the vapor bubble becomes small enough, mixing is promoted and the cold liquid is allowed to pass into the vapor bubble, collapsing it very quickly and/or dropping the pressure of the control volume rapidly if the bulk liquid is subcooled. During this transition, the pressure of the whole control volume drops rapidly from that of the vapor bubble to that of the saturation point of the cold liquid. The symptoms include small time steps, large pressure differences between the two tanks, and huge flowrates through the NULL connector. In the reverse transition (from all liquid to separated-phase), the symptoms can be similar if the transition occurs before sufficient mixing has occurred and the small tank has not grown appreciably. When transitions back to separated-phase logic occur, the volumes of the tanks can change instantly.

The user should note that the compliance of both tanks is set to zero in NONEQ1 if both are hard-filled (XL=0.0). If this is undesirable, the user may reset these compliances to small values (say 1.0E-3 divided by the system pressure) after NONEQ1, but *in other circumstances the user should avoid changes to tank or NULL simulation parameters if these elements are involved in a nonequilibrium simulation.*

If HLDLMP is used, the user should consider using HLDLMP on both tanks simultaneously, although this may not always be necessary.

Continuous Boil–off or Condensation While Separated—The significant addition of heat to the liquid tank (whether via the interface or from the walls) can cause continuous boiling in that tank. Similarly, heat extraction from the vapor tank can cause continuous condensation. If this action is dominant, the need for a nonequilibrium solution may disappear since the two tanks will tend to rapidly arrive at a state of thermal equilibrium. Otherwise, special modeling considerations come into play.

When too much vapor is generated in the liquid tank, or vice versa, the control volumes are instantaneously redrawn to move the "wrong" phase into the "right" tank. This action is controlled by a hysteresis band for stability. By default, the liquid tank is not allowed to rise beyond a quality of 1.0E–4 and the vapor tank is not allowed to fall below a quality of 0.99. These limits can be changed using the NONEQS routine described previously.

To increase the efficiency of solution, the amount of shifting and redrawing can be significantly reduced simply by twinning the NULL connector. This action automatically invokes special logic intended to expedite such analyses. One twin is used to model vaporization or condensation at the interface, and the other (otherwise unused) twin is used to move the "wrong" phase into the "right" tank at a continuous rate. This rate roughly corresponds to the boil–off or condensation rate such that the quality in the upstream tank (e.g., the liquid tank during boil–off) stays constant. Smoother, faster simulations generally result.

An Example of Nonequilibrium Simulation

The nonequilibrium behavior of a CPL reservoir is considered under simplified boundary conditions. The CPL reservoir is assumed to be a vertical cylinder 5 cm. in diameter and 50 cm long, and is being used in an ammonia system at 20°C. It is initially $^1/_4$ full of liquid by volume. The wall is assumed adiabatic for the purposes of demonstration. At time zero, fluid at 0°C is injected at a constant mass flow rate such that the reservoir becomes $^3/_4$ full in 60 seconds. The reservoir is then held for two minutes more, at which time liquid is extracted at the same rate as before for one minute, returning the fill level to $^1/_4$, where it is held for the final two minutes of the simulation. UVAP is held at -1.0 (solid conduction in the vapor) and ASUR corresponds to a disk-shaped interface. For comparison purposes, ULIQ will be parameterized from -1.0 (the minimum) to -1000.0 in order-of-magnitude increments. An analysis of a single perfectly mixed tank (corresponding to infinite ULIQ and UVAP) is also run.

Figure 6–2 shows the complete thermal response of the cylinder in the ULIQ=-1.0 case. Figure 6–3 shows the overall pressure response of all cases. The complete input file is included at the end of this section. In the ULIQ=-1.0 case, the heat transfer between the phases is so poor that virtually no change is evident when the reservoir is locked up. Clearly, the high temperatures ranges experienced in this case would increase the importance of the wall model in a real analysis. As the heat transfer rate (or degree of mixing in the liquid) increases, the peak pressure is reduced, and the reservoir approaches equilibrium faster.

The extremes of behavior in this example were purposeful. A long aspect ratio cylinder was chosen along with minimal heat transfer area, an adiabatic wall, and extremely fast fill and empty rates to exaggerate the nonequilibrium behavior. The user should not assume such extremes are typical. In normal pipe flow, the perfect-mixing assumption is usually acceptable.

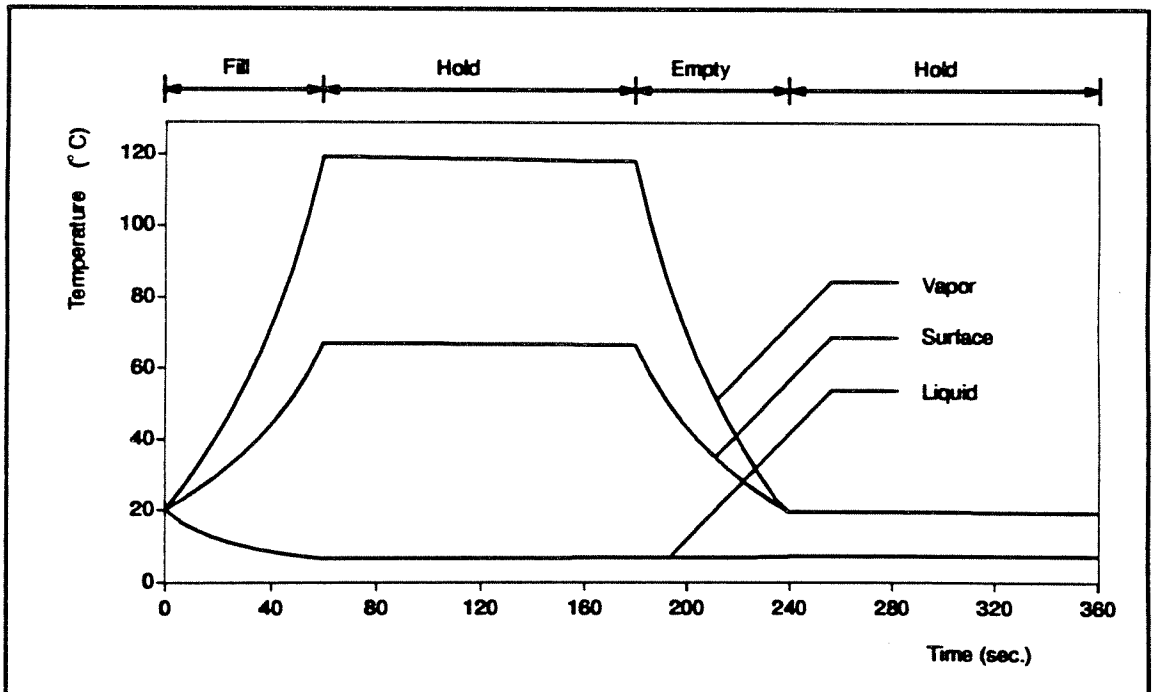


Figure 6–2 Temperature Response Given ULIQ = -1.0

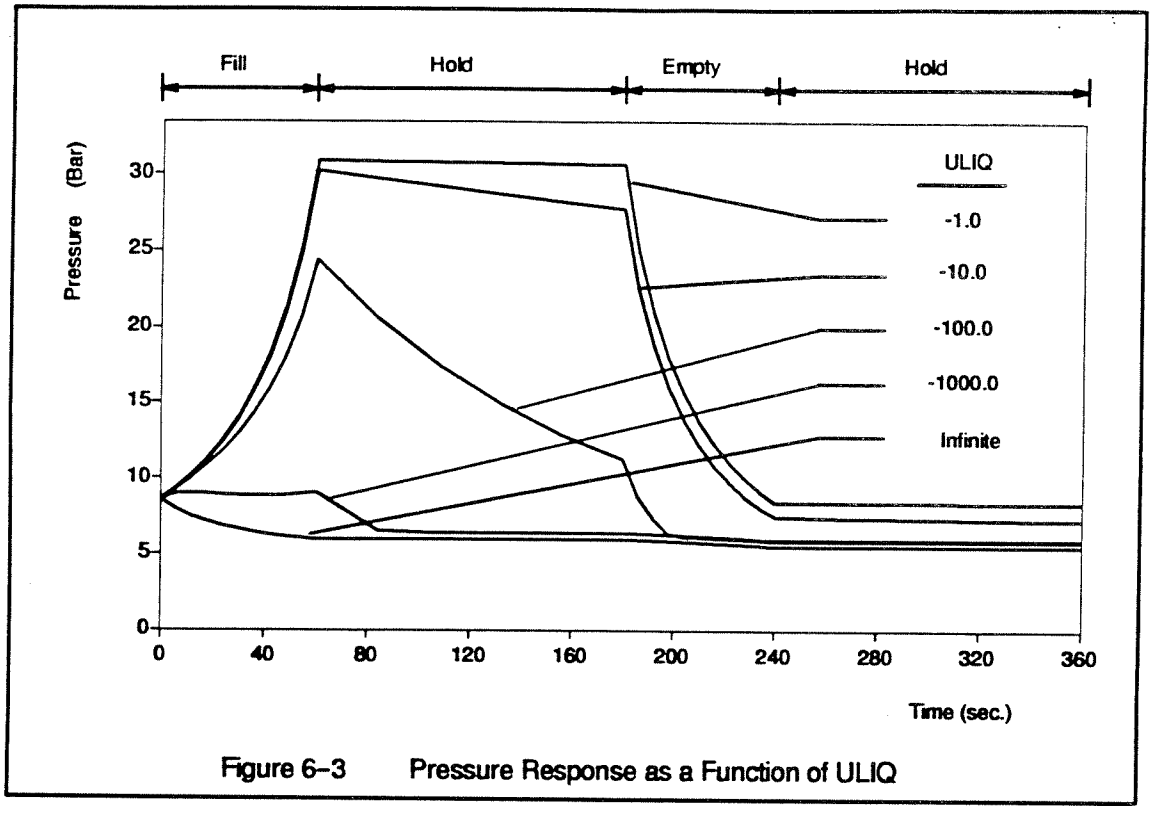


Figure 6–3 Pressure Response as a Function of ULIQ

As an exercise left to the user, try setting ULIQ "constant" at say -1.0 and varying ASUR parametrically to represent a tilted cylinder. More realistic boundary conditions such as the addition of a wall model and more realistic boundary conditions are also possible. The really adventuresome user may wish to try modifying the CPLGAS sample problem to include a non-equilibrium model of the reservoir during CPL start-up. The volume of that reservoir is approximately 280 cc, but its initial fill level, internal baffling, shape, mass, and heater controls are all unknown.

Sample Problem F contains another example of nonequilibrium simulation.

```
HEADER OPTIONS DATA
TITLE NONEQUILIBRIUM RESERVOIR - SECTION 6 EXAMPLE
C
       OUTPUT  = non.out
       USER1   = non.usr
C
HEADER FLOW DATA,RESERV,FID=717
C
LU DEF, TL=20.0
PA DEF, FR=0.0,          DH     = 0.10, TLEN    = 0.10
C LIQUID SIDE (AND WHOLE TANK DURING PERFECT MIXING RUN)
LU TANK,1,      XL=0.0, PL=0.0,         VOL=0.25*0.25*3.14*0.05*0.05*0.5
C VAPOR SIDE
LU TANK,2,      XL=1.0, PL=1.0E30,      VOL=0.75*0.25*3.14*0.05*0.05*0.5
C NULL CONNECTOR FOR NONEQ LOGIC
PA CONN,1,1,2,  GK=0.0, DEV     = NULL
C
LU PLEN,300,    XL=0.0,  PL=0.0, TL = 0.0       $ 0 C LIQUID SUPPLY
PA CONN,300,300,1,      STAT    = DLS,          $ INJECTOR/EXTRACTOR
                        DEV     = MFRSET
C
HEADER USER DATA,GLOBAL
       FILRAT  = 0.0                            $ FILL RATE (AND EMPTY RATE)
       ULIQ    = -1.0                           $ INITIAL LIQ SIDE HTC
       VOLRES  = 1.0                            $ RES VOLUME (OVERRIDEN LATER)
       ASUR    = 0.25*3.14*0.05*0.05            $ SURFACE AREA
HEADER CONTROL DATA, GLOBAL
       UID     = SI
       OUTPUT  = 1.0E10
       OUTPTF  = 6.0                            $ OUTPUT 6 SEC INTERVAL
       NLOOPS  = 50
HEADER OUTPUT CALLS, RESERV
       CALL LMPTAB('ALL')
       CALL PTHTAB('ALL')
       WRITE(NUSER1,11)TIMEN,TL1,TL2,VTS(PL1-PATMOS,RESERV.FI)+ABSZRO
     .          ,XL1,XL2,PL1/1.0D5
11     FORMAT(1X,1P,7G15.5)
C
HEADER OPERATIONS DATA
BUILDF NONEQ,RESERV
DEFMOD RESERV
C
       VOLRES          = VOL1 + VOL2
       CALL SAVPAR(NTEST)
C TRIPLE CURRENT LIQUID AMOUNT OVER 1 MIN
       FILRAT          = 2.0*DL1*VOL1/60.0
C
       DO 1 ITEST      = 1,5
C
C PULL BACK INITIAL CONDITIONS AND WRITE HEADER TO USER FILE
              CALL RESPAR(NTEST)
              WRITE(NUSER1,10)ULIQ
C
C IF SPECIAL FIFTH RUN (PERFECT MIXING) RESET TANK AND TURN OFF
C NONEQ LOGIC.
C
              IF(ITEST .EQ. 5)THEN
                     ATEST   = 0.75
                     ZTEST   = (1.0-ATEST)*DL1/(ATEST*DL2)
                     XTEST   = 1.0/(1.0 + ZTEST)
                     XTEST   = MIN(1.0,MAX(XTEST,0.0))
```

```
                    CALL CHGVOL('RESERV',1,VOLRES)
                    CALL CHGLMP('RESERV',1,'XL',XTEST,'PL')
                    HK300   = 0.0
                    GK300   = 0.0
                ENDIF
C       FILL
                TIMEO           = 0.0
                TIMEND          = 60.0
                SMFR300         = FILRAT
                CALL FWDBCK
C       HOLD
                SMFR300         = 0.0
                TIMEND          = TIMEND + 120.0
                CALL FWDBCK
C       EMPTY
                SMFR300         = -FILRAT
                TIMEND          = TIMEND + 60.0
                CALL FWDBCK
C       HOLD
                SMFR300         = 0.0
                TIMEND          = TIMEND + 120.0
                CALL FWDBCK
                ULIQ            = ULIQ*10.0
1       CONTINUE
C
10      FORMAT('1 RESULTS WHEN ULIQ = ',1pG13.5/
     . /5X,'TIMEN',T25,'TL',T40,'TV',T55,'TS',T70,'XL',T85
     . ,'XV',T100,'PL'/)
C
HEADER FLOGIC 0, RESERV
       IF(ITEST .LT. 5)
     .     CALL NONEQ0('RESERV',1,2,1)
C
HEADER FLOGIC 1, RESERV
       IF(ITEST .LT. 5)
     .     CALL NONEQ1('RESERV',1,2,1,ULIQ,-1.0,ASUR,1.0E30,VOLRES)
END OF DATA
```

6.9.6.6 Choked Flow Detection and Simulation Aids

Three routines are available to assist the user in detecting and simulating choked flow in individual paths: CHKCHP, CHKCHL, and CHOKER. The first (CHKCHP) is intended to keep an eye out for sonic throat limits (perhaps checking the whole subpanel for choking). The second routine (CHKCHL) can also perform that function for single paths but is intended to assist the user in modeling choking within FLOGIC blocks. The third (CHOKER) both detects choking and automatically simulates its effects on one or more connectors.

This section provides the background necessary to employ these routines correctly, describes the routines themselves, and finishes with an example.

These routines are invalid for 6000 fluids outside of the vapor region unless the VSOSFV function has been input or if the VDLC *and/or* the VSOSF functions have been input.

Choked flow calculations can only be performed for tubes and STUBE, LOSS, LOSS2, CHKVLV, and CTLVLV connectors. Furthermore, the routine CHOKER cannot be applied to tubes except within the solution routine FASTIC.

Background

The reader should keep in mind that FLUINT does not distinguish between static and dynamic thermodynamic states: in effect, all lump states are assumed to be static—velocities are assumed to be low and subsonic (typically less than Mach 0.3 to 0.4 for gas flows) such that the assumption of negligible kinetic energy changes is correct. However, this does not prevent accurate modeling of local choking within a valve or orifice, where the flowrate is constrained by the occurrence of choking at one point, but where the remainder of the flow is subsonic.

Where caution is required is in the modeling of straight ducts with no internal throats, nozzles, or radial mass injection. For flow to choke at the exit of such a duct, the velocities would have to rise gradually to the predicted critical value over enough length to potentially render the model invalid, since FLUINT neglects kinetic energy terms. At the very least, the critical flowrate will be in error since the assumption of a negligible upstream velocity will be invalid. This caution can probably be relaxed for most two–phase flows owing to the relatively small critical velocities typical of that region.

AFTH: Throat Area—For most K–factor loss elements and valves (LOSS, LOSS2, CTLVLV, and CHKVLV connectors), the minimum flow area is much less than the flow area that is used as the basis for dynamic head, AF. For sonic limit checks, this minimum throat area, AFTH, must be input as a separate parameter. In tubes and STUBE connectors with varying flow area, AF represents a characteristic flow area for friction, heat transfer, and inertia calculations and is normally greater than the value that should be used for choking calculations. If fact, if AF *doesn't* differ from the throat area and the flow within the duct is choked, then the assumption of low Mach number flow has already been violated in the upstream flow and the model is probably inappropriate.

AFTH may be input or defaulted for tubes and these five connectors. It represents the minimum flow area within the path, or at least the flow area that should be used for sonic limit checks. When choking calculations are performed, the fluid is assumed to be isentropically expanded from a stagnant (negligible velocity) upstream state down to the throat area, as described below. This assumption is applied even if AFTH was defaulted to AF. *Currently, choking calculations are the only use of AFTH.*

As a caution, the user might consider a value of AFTH that is even less that the actual minimum throat area for devices with abrupt reductions in flow area. For example, in the cases of sharp–edged contractions, inlets, and orifices, a *vena contracta* can form due to two–dimensional flow effects (separation), reducing the effective flow area to some fraction of the actual orifice opening. While it is very difficult to predict the size of the vena contracta, typical sizes are on the order of 80 to 95% of the actual opening area.

Two–Phase Sound Speed—Two methods are available for calculating the speed of sound in a two–phase mixture: Wallis' adiabatic method or Bursik's metastable method. The desired method is signaled by the integer argument METH: 1 for Wallis', 2 for Bursik's method.

Wallis' adiabatic method is simple and fast, but may overpredict the speed of sound by a little (typically a few percent, but potentially up to 20%). It predicts speed of sound by assuming a homogeneous mixture of two adiabatic phases, combining the phasic compressibilities like springs in series. Wallis also proposed another isothermal method which better predicts sound speeds in two–phase mixtures, but does not predict the right value in the limits of 100% vapor.

The second method (Bursik's, NASA TM 78810) is computationally slower, but combines the best features of Wallis' two methods: it is a better estimation at low qualities, and it predicts the correct value in the limit of 100% vapor. However, the user should note that, like almost any two–phase prediction, there is no definitive method nor exact solution. In other words, either method is probably as accurate as the other, since neither take into account flow regime variations and other effects that are known to be important.

Expansion Process and Critical Flowrate Estimation—In determining the critical throat flowrate, the current upstream state is assumed to represent static or stagnation (zero velocity, or at least negligible kinetic energy) conditions: $AFTH << AF_{inlet}$. AFTH, the throat area, is used to compute the choked mass flowrate *assuming that the flow is isentropically expanded from the zero velocity upstream state to a critical throat state.* Because AFTH defaults to AF, and because the throat velocity is frequently much larger than the characteristic head velocity for valves and orifices, the user is strongly encouraged to consider an explicit input for AFTH if choking is a concern.

The assumption of isentropic expansion from inlet to the throat is a very common assumption that is adequate for single–phase flows. In two–phase flows, this assumption is combined with that of homogeneous equilibrium flow. The resulting methods predict the throat pressure with reasonable accuracy, but can underestimate the critical flow rate at low throat qualities. This approach is reasonable, relatively cheap to implement, robust, and even conservative. More realistic expansion processes* take into account the fact that the proportions of liquid and vapor do not change significantly during the rapid expansion, while the local rate of interphase

* R. E. Henry and H. K. Fauske, *The Two–Phase Critical Flow of One–Component Mixtures in Nozzles, Orifices, and Short Tubes.* Journal of Heat Transfer, May 1971, Pp 179–187.

heat transfer is very large because the vapor cools much faster than does liquid, resulting in a polytropic expansion whose characteristics must be estimated with empirical correlations. The user should keep in mind such additional uncertainties when choosing between the two methods provided for speeds of sound: if the user has no other preference or information, Wallis' method (METH=1) should be used because it is cheaper and it tends to overestimate the sound speed, perhaps helping to balance the underestimations of the simplified expansion process. In fact, in many tests Wallis' method predicts a slightly (5 to 10%) smaller critical flowrate in liquid and low–quality regions than does Bursik's method because of the importance of the expansion process compared to the speed of sound estimation.

Subcooled single phase liquid almost always expands into a two–phase state at the throat: this is the basis of operation for the cavitating venturi. Otherwise, at extremely high pressures, liquid in FLUINT does not expand at all due to the persistence of the incompressible liquid assumption: it merely speeds up at constant saturation density until it reaches (perhaps) its sonic velocity.

The inclusion of the expansion process results in critical flowrates that are at usually at least a factor of two lower than value calculated assuming that the flow has already been expanded, accelerated, or otherwise heated up to the sonic condition in a constant area duct (i.e., the flow area times the upstream speed of sound times the upstream density). Since liquids almost always expand into a two–phase throat, the difference can easily be two or more orders of magnitude. If for some reason the user wishes to neglect the expansion process (perhaps it has already been explicitly included in the model using the AC factors on upstream paths), then direct calls to the speed of sound routines should be used instead. The cavitating venturi example provided below helps clarify this distinction.

For twinned (slip flow) paths, the critical throat state is calculated based on homogeneous isentropic expansion from the upstream state, and the resulting speed of sound is compared against *both* phasic velocities (calculated using the throat area but apportioned according to the upstream void fraction). A warning is produced if *either* twin exceeds the limit, or if the total flowrate exceeds the returned value of the total critical flowrate. The latter is calculated assuming that *both* phases are sonically limited at the aforementioned hypothetical homogeneous throat state. The returned flowrate has less meaning for twins than for single paths, especially the counterflow case (which should be exceedingly rare if not impossible for such high velocities). While the above assumptions are consistent and blend with the homogeneous assumption, the user is cautioned that they are somewhat capricious since little basis exists to justify them.

Subroutine Name: CHKCHP

Description: Intended to keep watch on all or part of a submodel for localized choking, CHKCHP prints out a warning to the output file when a sonic limit is encountered. No other action is taken; the user must then decide what, if any, model or logic changes are required. CHKCHP may be used to keep an eye on single paths (or single pairs of twinned paths), or, by setting the path ID to zero, it may be used to watch all appropriate paths.

CHKCHP can be called from any logic block, although FLOGIC 2 is probably the most appropriate block for almost all analyses.

The critical flowrate is calculated based on the speed of sound in the throat (defined by AFTH), assuming isentropic expansion to a homogeneous equilibrium state from a zero–velocity upstream state.

Restrictions: If all paths are to be checked, CHKCHP ignores connector devices with no defined flow or throat areas: MFRSET, VFRSET, PUMP, VPUMP, CAPIL, and NULL. If one of these types of path is named specifically, the routine will cause an abort. For twinned tubes and STUBE connectors, input the primary path only.

Calling Sequences:

```
CALL CHKCHP(fsmn, pathid, METH)
```

where:

fsmn ..... fluid submodel name, single quotes
pathid ..... ID of path to check. Zero means check all paths in submodel. For twins, input ID of primary twin.
METH ..... method to use for two–phase speed of sound if encountered:
1 = Wallis' adiabatic
2 = Bursik's metastable

The last argument, METH, is the same as that in VSOSFV: the identifier for the method to be used for two–phase sound speed if two–phase flow is ever encountered.

Examples:

```
CALL CHKCHP('GLUG',0,1)        $ CHECKS ALL PATHS, WALLIS'
CALL CHKCHP('GLUG',10,2)       $ CHECKS PATH #10, BURSIK'S
```

Subroutine Name: CHKCHL

Description: If the user knows ahead of time which paths will or might be choked (perhaps from previous runs using CHKCHP), the routine CHKCHL is intended to provide the user with the information necessary to simulate locally choked flow for a single path. (Multiple calls, of course, can be used to cover more than one potential choking site.) For each path called, the program returns the value of the sonically limited flowrate as well as an integer flag that is zero if the path is currently choked, and zero if not. No other action is taken; the user must then decide what, if any, model changes are required. CHKCHL may be used on single paths or single pairs of twinned paths. Unlike CHKCHP, it *cannot* be used to watch all appropriate paths.

The critical flowrate is calculated based on the speed of sound in the throat (defined by AFTH), assuming isentropic expansion to a homogeneous equilibrium state from a zero–velocity upstream state.

CHKCHL can be called from any logic block, although FLOGIC 0 is probably the most appropriate block for almost all analyses.

Guidance: Unless the path is a tube, the user is referred to the routine CHOKER, which is easier to apply.

Restrictions: Results in an abort if used for connector devices with no defined flow or throat areas: MFRSET, VFRSET, PUMP, VPUMP, CAPIL, and NULL. For twinned tubes and STUBE connectors, input the primary path only.

Calling Sequences:

```
CALL CHKCHL(fsmn, pathid, METH, ISCHK, FRC)
```

where:

fsmn	fluid submodel name, single quotes
pathid	ID of path to check. For twins, input ID of primary twin. Zero is invalid and will result in an abort.
METH	method to use for two–phase speed of sound if encountered: 1 = Wallis' adiabatic 2 = Bursik's metastable
ISCHK	returned integer flag: 0 = path is *not* choked 1 = path *is* choked
FRC	returned choked flowrate (calculated even if path is not choked)

The argument METH is the same as that in VSOSFV: the identifier for the method to be used for two–phase sound speed if two–phase flow is ever encountered.

Examples:

```
      CALL CHKCHL('GLUG',131,1,ITEST,FTEST)
C
      DO 100 MTEST = 1,10
100       CALL CHKCHL('GLUG',NA(2+MTEST),2,NA(3+MTEST),A(4+MTEST))
```

In the first example above, a single path (numbered 131) is being checked for a sonic limit. Later logic can check the value of ITEST to see if choking was indeed found, and if so the choked flowrate is available in FTEST for modeling purposes (see next subsection). If any two–phase flow is encountered, Wallis' method is used to predict speed of sound.

In the second example, the user has set up three arrays, two integer and one real, to automate the logic for a set of 10 paths. The first (array #2) contains prestored path ID numbers, the second and third (#3 and #4) are reserved to contain the returned values of ISCHK and FRC for each of the ten paths.

Subroutine Name: CHOKER

Description: Like CHKCHP, this routine keeps watch on all or part of a submodel for localized choking. Unlike CHKCHP or CHKCHL, it then proceeds to simulate the effects of choking anyplace it is detected. No other model or logic changes are required.

CHKCHP may be used to keep an eye on single paths (or single pairs of twinned paths), or, by setting the path ID to zero, it may be used to watch all appropriate paths (see Restrictions below).

CHKCHP can only be called from FLOGIC 1. It does not apply to tubes (except within FASTIC).

The critical flowrate is calculated based on the speed of sound in the throat (defined by AFTH), assuming isentropic expansion to a homogeneous equilibrium state from a zero–velocity upstream state.

Restrictions: ***CHOKER must be called from FLOGIC 1 only.*** *It does not apply to tubes (except in FASTIC).* If all paths are to be checked, CHOKER ignores tubes (except in FASTIC) connector devices with no defined flow or throat areas: MFRSET, VFRSET, PUMP, VPUMP, CAPIL, and NULL. If one of these types of path is named specifically, the routine will cause an abort. For twinned connectors, input the primary path only.

Calling Sequences:

```
CALL CHOKER(fsmn, pathid, METH, HYST)
```

where:

fsmn ..... fluid submodel name, single quotes
pathid ..... ID of path to check. Zero means check all relevant paths in submodel. For twins, input ID of primary twin.
METH ..... method to use for two–phase speed of sound if encountered:
1 = Wallis' adiabatic
2 = Bursik's metastable
hyst ..... hysteresis to apply to critical flowrate calculation, must be greater than or equal to zero and less than one.

The argument METH is the same as that in VSOSFV: the identifier for the method to be used for two–phase sound speed if two–phase flow is ever encountered.

The last argument, HYST, defines the hysteresis band to apply to the critical flowrate calculation. It should be left as zero unless stability problems are suspected (e.g., a path seems to oscillate between choked and unchoked states). Otherwise, a value of 0.01 implies an uncertainty in the critical flow calculation of 1%, with the current state (e.g., choked or not) continued within the band of uncertainty. Values greater than 0.1 (10%) should not be used, although any value less than one (and greater than or equal to zero) is legal.

If twinned tubes (in FASTIC) or STUBEs become choked, then the homogeneous solution is applied internally even though slip flow will appear to continue.

Guidance: There is no way to determine whether or not a path is choked using CHOKER alone. If this information is required, use the CHKCHP or CHKCHL routines in combination with CHOKER, allowing for the effects of hysteresis if any is applied.

Examples:

```
CALL CHOKER('BM',0,1,0.0)      $ OPERATES ON ALL PATHS, WALLIS'
CALL CHOKER('XLX',10,2,0.01)   $ OPERATES ON PATH #10, BURSIK'S
```

Modeling Choked Flow

Choked flow represents a flowrate constraint on the system. The flowrate through a choked path cannot exceed the sonic limit, and is completely independent of downstream pressure or other conditions. However, it *is* dependent on upstream conditions. For example, the critical flowrate of a perfect gas through a choked orifice is proportional to the absolute upstream static pressure, and to the inverse square root of the absolute upstream static temperature.

The easiest way to model such choking is to use the CHOKER routine. Unfortunately, this routine does not apply to tubes (except in FASTIC).

An alternative to CHOKER is to use an MFRSET connector, whose flowrate (SMFR) is set to the sonic limit, as returned by CHKCHL. If the flow only intermittently choked, or if the condition is unknown and the user desires to cover both cases, then the MFRSET can be used in parallel with the real element (tube, STUBE, or LOSS–type connector). When choked, the MFRSET can be used, and when not, the normal connector can be used. An important question is: If the flow is currently choked and the MFRSET is controlling the flowrate, how can one tell how much flow would go through the device if it *weren't* choked? In other words, how does one know if the flow is still choked? The answer is: the user must either make that calculation in logic blocks, or let FLUINT make that calculation without affecting the rest of the network. The key to the latter method is the use of duplication factors.

To illustrate, take the following example of blowing down a tank through a line which has a valve at the end. While the flow is initially choked, as the upstream pressure drops down to where the flow is subsonic through the valve. Assume that the FLOW DATA block for this submodel named "BLOW" described an MFRSET connector #10 in parallel with a LOSS connector #20 whose K–factor is 100. The following logic might be used:

```
HEADER FLOW DATA, BLOW, FID= ...
...
PA DEF, FR= 100.0, DH = 0.01
PA CONN,10,1,2, DEV = MFRSET
PA CONN,20,1,2, DEV = LOSS, FK=100.0, AFTH = 1.0E-5
...
HEADER FLOGIC 0, BLOW
      CALL CHKCHL('BLOW',20,1,ITEST,FTEST)
      IF(ITEST .EQ. 1)THEN
            SMFR10      = FTEST
            DUPI20      = 0.0
            DUPJ20      = 0.0
      ELSE
            SMFR10      = 0.0
            DUPI20      = 1.0
            DUPJ20      = 1.0
      ENDIF
```

In other words, when choked, the MFRSET is set to the choked flowrate and the LOSS is DUPed out of the network. When not choked, the MFRSET turned off and the LOSS is reinstated. The important point is that the LOSS connector itself continues to predict the flowrate

through it at all times, unaware that it has no effect on the rest of the network while its DUP factors are zero. If instead, a CTLVLV had been used and it was turned off by setting FK negative, then CHKCHL would have always returned ITEST=0 (i.e., not choked) *since the flowrate through the CTLVLV would have remained at zero.* Therefore, DUP factors can be used to force FLUINT to run a 'sideline' calculation.

The equivalent (and preferred) usage of the CHOKER routine for the above case is as follows:

```
HEADER FLOW DATA, BLOW, FID= ...
...
PA DEF, FR= 100.0, DH = 0.01
PA CONN,20,1,2, DEV = LOSS, FK=100.0, AFTH = 1.0E-5
...
HEADER FLOGIC 1, BLOW
      CALL CHOKER('BLOW',20,1,0.0)
```

In some models, the stark differences in behavior between the choked and no–choked states may cause stability problems, with the model flipping back and forth between these two states. Slightly more complicated logic can be used to add hysteresis to the decision, which is usually enough to avoid the problem. An example of such logic applied to the above example is listed below, which uses a ±10% deadband on the sonic limit prediction:

```
HEADER FLOGIC 0, BLOW
      CALL CHKCHL('BLOW',20,1,ITEST,FTEST)
      IF(ITEST .NE. MTEST)THEN
            IF(MTEST .EQ. 1)THEN
                  IF(ABS(FR20) .LT. 0.9*FTEST) MTEST = 0
            ELSE
                  IF(ABS(FR20) .GT. 1.1*FTEST) MTEST = 1
            ENDIF
      ENDIF
      IF(MTEST .EQ. 1)THEN
            SMFR10      = FTEST
            DUPI20      = 0.0
            DUPJ20      = 0.0
      ELSE
            SMFR10      = 0.0
            DUPI20      = 1.0
            DUPJ20      = 1.0
      ENDIF
```

In the above example a new variable, MTEST, is used to store the current or last state of the device according the same rules as ISCHK. *Because of this use of hysteresis, it is the user's responsibility to input initial conditions (including MTEST) that are completely consistent with either a choked or unchoked initial condition.*

Once again, the equivalent (and preferred) usage of the CHOKER routine for the above case is as follows (all that changes is the last argument, the hysteresis factor):

```
HEADER FLOW DATA, BLOW, FID= ...
...
PA DEF, FR= 100.0, DH = 0.01
PA CONN,20,1,2, DEV = LOSS, FK=100.0, AFTH = 1.0E-5
...
HEADER FLOGIC 1, BLOW
      CALL CHOKER('BLOW',20,1,0.1)
```

The use of MFRSET connectors to simulate choking can be troublesome in some models because of their unforgiving control of the flowrate. Unless elements other than junctions and rigid liquid–filled tanks are used in series with the choking path, then the flowrate in the line can be overconstrained by other MFRSETs, VFRSETs, shut valves, or primed CAPILs or CAPPMPs in series with the choked path. (Such overconstraining does not occur with the use of the CHOKER routine.) Some amount of forgiveness can be gained by adding yet a third connector in series: a LOSS–type connector with a large K–factor, or by modeling the choked connector with a NULL connector with nonzero GK (and perhaps nonzero EI)—a solution for advanced users only. Unfortunately, none of these methods implicitly takes into account the invariance of choked flowrate with downstream pressure. During each solution step, a change in conditions in either lump will have an equal effect.

For twinned tubes or STUBE connectors, it is likely that only one phase (most likely vapor) will be choked. Modeling methods have not been established for this special case. The user should probably revert to a homogeneous assumption using GOHOMO, which enables the above methods. Within CHOKER, such homogeneous assumptions are applied internally when twinned paths become choked.

Example: Cavitating Venturi

To help illustrate some of the subtleties of choked flow modeling, a specific example will be worked using 'old methods,' CHKCHL, and CHOKER. Some of this text and data appeared in *Applications of a General Thermal/Hydraulic Simulation Tool,* AIAA–89–1754, June 1989, by B. Cullimore.

Component Description—The cavitating venturi is a flow control device commonly used in liquid propulsion systems. It is also baselined in the space station thermal bus. Similar in construction to a conventional venturi, the flowrate in a cavitating venturi is limited because the throat pressure cannot decrease below the saturation pressure—the liquid begins to flash in the throat and quickly becomes choked because of the sharp reduction in speed of sound associated with even a tiny amount of vapor present. When choked, the flowrate becomes independent of the downstream conditions, and is only slightly affected by the upstream pressure (or the degree of subcooling).

The actual processes in the diffuser section are poorly understood owing to the nature of the vena contracta, the lack of thermodynamic equilibrium, and the unsteady two–dimensional flow patterns. Physically, liquid jets through the center of a vapor bubble which forms around the diffuser wall. The vapor recondenses (probably in a standing shock wave or hydraulic jump) and liquid exits the venturi. Typically, cavitation cannot occur until the back pressure (roughly equal to the throat pressure) is less than about 85% of the upstream pressure. This upper limit on back pressure is relatively easily calculated. If the back pressure is too low, the cavitation bubble extends beyond the diffuser: two–phase flow results in the outlet. This lower limit is less predictable than the lower limit but is fortunately rare in real applications.

Randall* shows a cross section of a typical cavitating venturi for a 1 inch (2.54 cm) ID line (Figure 6–4), along with results of performance calibration testing (Figure 6–5). In Figure 6–5, the lower line represents test data wherein the upstream pressure was held at 600 psia (4.14 MPa) while the back pressure was lowered until the lower limit was reached at atmospheric pressure. The upper line represents the subsequent lowering of the upstream pressure. The upper regions represent cavitation conditions.

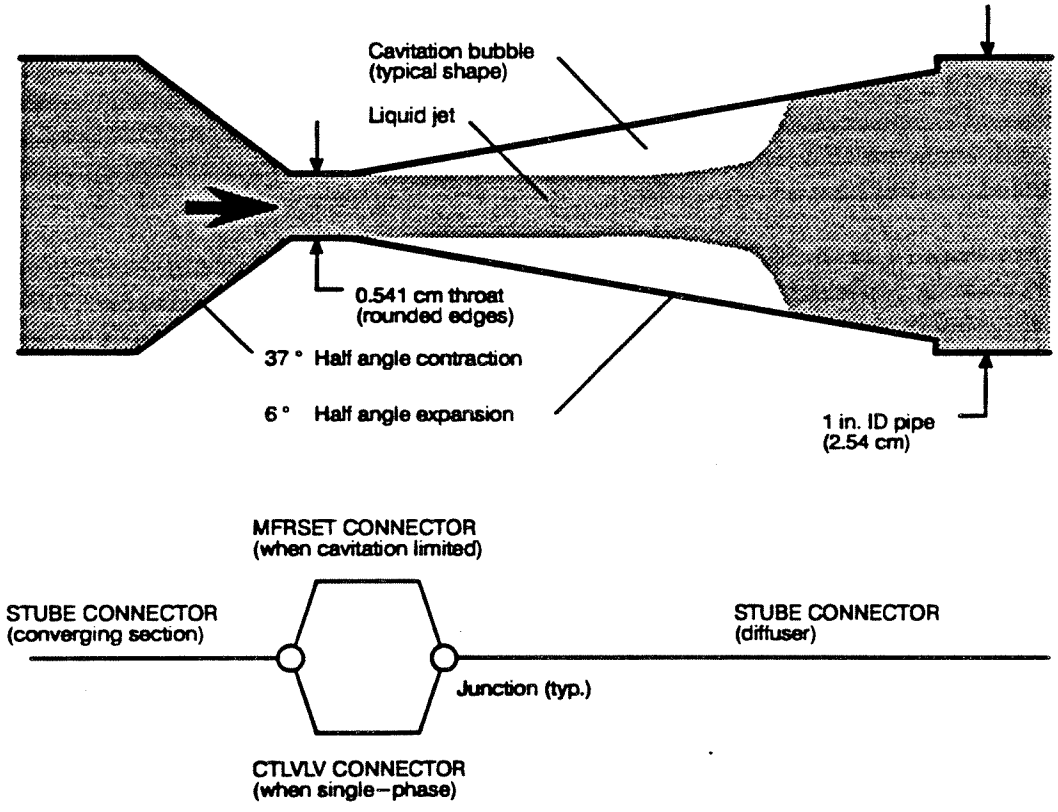


Figure 6–4 Cavitating Venturi and Corresponding Schematic for Old Modeling Approach

* L.N. Randall, *Rocket Applications of the Cavitating Venturi*, Proceedings of the American Rocket Society, Toronto, Canada, June 1951.

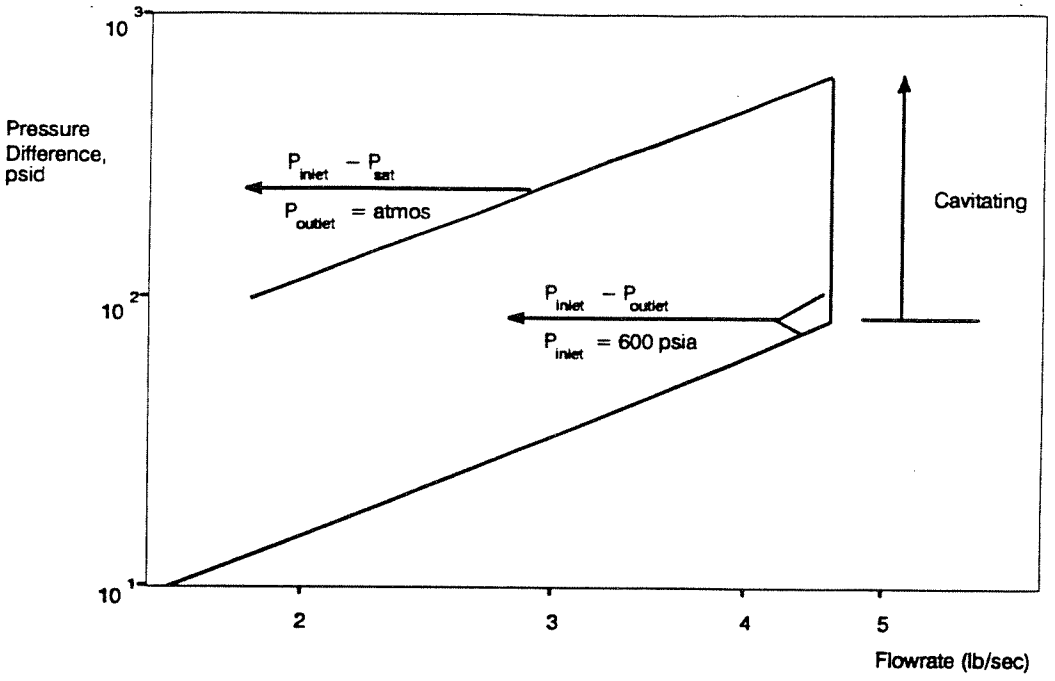


Figure 6-5 Cavitating Venturi Calibration

Comparison (Former) Modeling Approach—The venturi throat is represented by parallel connectors: an MFRSET and a CTLVLV. When the unit is cavitating (as indicated by the quality at the throat *inlet*), the CTLVLV is closed and the MFRSET set point flowrate is adjusted in the user logic blocks to limit the flowrate according to the inlet conditions. In other words, the flowrate is calculated such that the inlet expansion process exactly reaches the saturation point at the throat. When the unit ceases to cavitate (as indicated by the quality at the throat *outlet*), the MFRSET connector is closed by setting its flowrate to zero, and the CTLVLV is opened to a small K-factor such that the pressure drop through the nonphysical "valve" is negligible.

The converging and diverging portions of the venturi are represented by STUBE connectors. The AC and FK coefficients for these devices are calculated in user logic to model these sections. Normally, in plain duct models the AC and FC factors are "internal" variables calculated by the program and available to the user for inspection or modification. In this case, the AC factor is used to model the area change:

$$AC = \frac{1}{2\rho}\left(\frac{1}{AF_i^2} - \frac{1}{AF_j^2}\right) \qquad \text{(AREA CHANGE)}$$

where ρ is the density, and AF_i and AF_j are the defined inlet and outlet flow areas, respectively. For the inlet STUBE, AF_j is the throat area.

Unfortunately, K–factors for converging and diverging sections are less easily defined, and hence were to fit the experimental data. For the venturi in question, handbooks suggest K–factors based on the throat area of 0.08 and about 0.3 for the converging and diverging sections, respectively. The actual losses appear to be much less: about 0.025 and 0.12. Rounding at the throat is at least partially responsible for this difference.

The FK in the diverging section was assumed constant, even in the cavitating regime. This is appropriate not only because better estimates would be extremely difficult and uncertain, but because the actual loss in the diverging section becomes completely negligible compared to the loss through the MFRSET connector. While the location of the MFRSET and diverging STUBE may not be physically realistic (compared to the actual position of the shock wave or hydraulic jump), the total behavior of the cavitating venturi is realistically represented from a system perspective.

There are some assumptions inherent in this model. First the flowrate is assumed to always be positive. More extensive logic could be included for two–way operation if desired, including different FK values. Second, the FK values are assumed constant for all flowrates, which implies fully developed turbulent flow. Third, the model does not include the lower operating limit: the vapor bubble is assumed to be always contained within the venturi. While the program would not balk at two–phase flow in the outlet, the FK term in the diverging section would be too low, and the corresponding AC term would be in error since only area change was considered; density changes (spatial accelerations) could also be added to better cover this case.

Another excellent assumption is that the throat state is exactly saturated (instead of slightly two–phase), and that the sonic velocity is therefore largely irrelevant since *any* cavitation is presumed to be enough to choke the flow. Otherwise, a difficult iteration would need to be performed to find the exact throat state and corresponding sound speed (using VSOSFV), and very little difference would have been noticed in the results. CHKCHP and CHKCHL *do* perform this iteration along with an internal model of the inlet expansion process.

Unlike examples of choked flow modeling in earlier sections using DUP factors, this model bases the choke/no–choke decision on the throat outlet quality, which naturally contains a slight hysteresis due to the low level of flashing across the CTLVLV itself: the venturi chokes when the throat outlet becomes two–phase, and returns to the unchoked state when zero quality reappears at that point.

Modeling Approach Using CHKCHL—If one is willing to make the excellent assumption that the pressure losses in the inlet and diffuser sections are negligible *while the flow is choked*, then a much simpler model can be used by employing the CHKCHL routine, as will be described in this subsection.

The preceding model is replaced by two paths in parallel: a single STUBE connector and a single MFRSET connector.

The STUBE represents the entire venturi in the unchoked state. Because there is no net area change, AC is zero. Also, FK is set equal to the sum 0.025 + 0.12, representing losses in both the converging and diverging sections. Like the preceding model, AF is set to the throat area since this is the area upon which those K–factors are based. For this reason, the throat area AFTH is defaulted to AF.

As with the previous model, the MFRSET connector is used to represent the cavitating venturi in the choked state; otherwise, it is zeroed out by setting SMFR=0.0. When the venturi is cavitating, the STUBE is zeroed out by setting DUPI=DUPJ=0.0, according to earlier examples, such that CHKCHL may be used to govern both the choke/no–choke decision as well as to determine the choked flowrate.

CHKCHL dutifully calculates the delicate balance at the throat, assuming isentropic expansion from the inlet state. Note that in the previous model the expansion process was not assumed to start from a zero–velocity state: the implicit CHKCHL assumption is that $(1/AF_{inlet})^2 << (1/AFTH)^2$, or in this case $1 << 486$. The fact that AF is coincidentally equal to AFTH is irrelevant from CHKCHL's viewpoint.

The resulting model produces nearly identical results to that of the 'old' method.

Modeling Approach Using CHOKER—The above approach using the CHKCHL routine can be simplified even further with the CHOKER routine.

Using CHOKER, the MFRSET becomes unnecessary: the STUBE represents the entire venturi in the both the choked and unchoked states. Logical checks and manipulations are therefore unnecessary; only the call to CHOKER is required in FLOGIC 1.

Although the use of a tube is not appropriate in this example (since the inertia of the venturi is negligible for most simulation requirements), if a tube *had* been required the CHOKER routine would not have been applicable outside of FASTIC. In such a case, the CHKCHL method could still be applied.

6.9.6.7 Modeling Water Hammer and Acoustic Waves

Using tanks, tubes, and an appropriately constrained time step, SINDA/FLUINT may be used to accurately simulate propagation of acoustic waves* in liquids and gases. Excellent comparisons have been made with analytic solutions, test results, and even other codes that specialize in the calculation of such phenomena. This section describes the methods that should be applied along with two utility routines intended to facilitate such modeling. Refer also to Section 4.7.11.4.

Modeling fast hydrodynamic events such as water hammer and pressure waves in gases implicitly requires the use of high temporal resolution fluid elements such as tanks and tubes. This does not mean that the entire model need be built from these elements. Rather, all significant volumes must be accounted for with tanks, and all significant lengths must be accounted for with tubes. Junctions and connectors may also be used as is appropriate given their instantaneous (massless, inertialess) nature.

Results are largely insensitive to the number of tanks and tubes (e.g., the spatial resolution). As long as no significant property (e.g., density) gradients exist in a line, relatively coarse resolution is adequate. This is especially true for centered HX and LINE macros, which should be used in preference to up– or downstream options.

On the other hand, results can be very sensitive to the time step chosen. FLUINT was specifically designed to avoid the necessity of taking time steps small enough to resolve such fast transient events. Because of its implicit solution, it will tend to smear out such waveforms and perhaps miss them altogether unless the time step is specifically constrained. Loss of such high frequency phenomena is no concern when the problem at hand focuses on lower frequency phenomena such as heat transfer transients. However, when the design concern is flow–induced loads, then accurate calculation of peak pressures requires that the time step be limited.

Comparisons with analytic solutions have yielded a *recommended* maximum time step to apply when accurate calculation of such waveforms and peak pressures are required: $[L/(2a)]_{min}$, where L is the length of a segment and a is the local speed of sound in that segment. Once a spatial resolution has been chosen, a maximum time step exists above which accuracy is lost. A routine ("WAVLIM") is available to calculate this limiting time step. *The user should note that even smaller time steps may be required in some models, and that either a safety factor should be applied or an assessment should be performed of the sensitivity of results to the chosen time step.*

Liquids are assumed to be incompressible, with infinite sound speeds. The slight compressibility that most liquids exhibit can be accurately modeled with the compliance factor (Section 4.7.11.4). A routine ("COMPLQ") is available to facilitate this calculation. Note that the compressibility of the fluid need not be added unless the tank contains 100% liquid (zero quality), since the program already considers the fluid to be compressible when any vapor is present.

In addition to the compressibility of the fluid itself, additional compliances may be required to model wall stiffness, trapped gases, etc. Such compliances will reduce the sound speed.

---

* excluding shock waves

Subroutine Name: COMPLQ

Description: This routine sets the compliance of all liquid–filled tanks in a fluid submodel to be equal to the compressibility of the liquid. In other words, since liquid always assumed to be incompressible, the slight compressibility exhibited by most liquids can be modeled as a slight deflection of the control volume boundary. This routine is intended as an aid in modeling water hammer, or for easily adding realistic compliances into a system simulation.

COMPLQ should be called from within FLOGIC 0. If WAVLIM is also used, it should be called *after* COMPLQ.

Restrictions: This routine overwrites the current value of COMP for all tanks in the model, regardless of their thermodynamic state. Therefore, it should not be used in combination with routines that manipulate the COMP value (e.g., NONEQ1, BELACC).

The liquid phase is assumed incompressible (zero compliance) if the fluid is a 9000 series user fluid, or if it is a 6000 series user fluid in which neither VDLC nor VSOSF have been supplied. No warnings are produced in these cases.

Calling Sequences:

```
CALL COMPLQ(fsmn,cmpadd)
```

where:

fsmn ..... fluid submodel name, single quotes
cmpadd..... additional compliance to be added to all tanks

Normally, cmpadd will be zero although it may be used to specify container rigidity, included gases, etc. The compliance of all tanks will be set to cmpadd. An additional compliance (corresponding to the liquid compressibility) will be added for all tanks that are currently completely filled with liquid.

Example:

```
CALL COMPLQ('FLOSS',0.0)
```

Subroutine Name: WAVLIM

Description: Calculates the maximum recommended time step to be used if the capture of fast hydrodynamic transient events is required. This limit is calculated as one half the ratio of the smallest tube length in the submodel to the largest sound speed in all tanks in the submodel: $TLEN_{min}/(2*a_{max})$. This is a recommendation only; even smaller time steps may be necessary in some cases.

For liquid–filled tanks, the sound speed is calculated on the basis of user–supplied compliances. For tanks containing any vapor, the sound speed is calculated on the basis of the fluid sound speed plus the effects of any additional compliances.

WAVLIM should be called from within FLOGIC 0. If COMPLQ is also used, it should be called *before* WAVLIM.

Restrictions: An abort will result if both tanks and tubes are not present, or if all tanks are noncompliant and contain liquid.

Calling Sequences:

```
CALL WAVLIM(fsmn,dtmax)
```

where:

fsmn ..... fluid submodel name, single quotes
dtmax ..... returned maximum time step for fast hydrodynamic transients

Examples:

```
CALL WAVLIM('SURFS',FLOSS.DTMAXF)

CALL WAVLIM('UP',DTEST)
UP.DTMAXF = 0.2*DTEST
```

7. EXTERNAL PROGRAM LIBRARY

In addition to the preprocessor executable and the processor library, SINDA/FLUINT includes three separate companion programs that are described in the following sections. These programs are:

1). EXPLOT, a graphics post-processor for the VAX version. Reads SAVE files and produces plots. Requires DISSPLA graphics software.

2). DATA_ONLY, a nongraphical post-processor. Reads SAVE files and produces tables.

3). RAPPR, a pre-preprocessor that creates fast alternative fluid descriptions (i.e., FPROP DATA blocks) from standard library descriptions.

Although still available, EXPLOT and DATA_ONLY have been rendered obsolete by the creation of SINAPS (SINDA Application Programming System), an interactive graphical pre- and postprocessor for SINDA/FLUINT. A brief introduction to SINAPS is given in the preface to this manual.

7.1 EXPLOT: External Plotting

7.1.1 Introduction

EXPLOT is the external plotting package for the VAX version of SINDA/FLUINT. (For other machines, it has been replaced by SINAPS.) It allows the user to plot data saved from within SINDA/FLUINT runs. EXPLOT is an interactive interface to DISSPLA, but allows batch plotting also.

The main features of EXPLOT are page and graph control. The user can specify page size and orientation as well as the number of plots per page. The user has control of graph titles, page titles, axis labels, the number of axes, curve legends, the number of curves on a plot, and the content of each curve. The plotted data can come from up to 20 different SAVE files. All input commands are saved on a user specified output file that can be altered for future runs.

The user input is divided into several sections. Each section is presented in this document as an annotated EXPLOT session. There is also a brief mode of operation for experienced users which is discussed in Section 7.1.7.

7.1.2 General Principles Of Operation

The user interface with EXPLOT is meant to be self explanatory. The program asks a series of questions. Included in each question is the default answer, which is invoked by simply pressing a carriage return. In this manual, carriage returns will be indicated by [RET].

EXPLOT uses six files. The first three files are input files, including the mandatory *SAVE file* from a SINDA/FLUINT run. (Restart files may be used equivalently.) The other two input files are the optional *command* and *brief files*. When the user answers the questions that EXPLOT asks, the responses are saved on a *response file*. At a later session these responses can be read in as a command file. There is also an optional *brief file* which provides default responses to a number of questions. The brief file is explained in Section 7.1.7.

Three are three output files: the response file mentioned above and two other files named EXPLOT.MIN_MAX and EXPLOT.DATA_POINTS. The file EXPLOT.MIN_MAX contains the maximum and minimum data values for each plotted curve along with annotation identifying the curve. EXPLOT.DATA_POINTS contains all the points on the curve (time and data) along with annotation identifying the curve.

7.1.3 Terminal Selection

The first question establishes the type of terminal being used. If the user is at a graphics terminal, a menu will appear to determine the type of graphics terminal and the plots will appear at the terminal as each page of information is completed. If the user is not at a graphics terminal, he will be asked if he wishes to produce a DISSPLA metafile as each page of information is completed. If the user answers "no" to this question, then only an output file containing the responses to the questions will be produced.

Terminal selection example:

```
ARE YOU AT A GRAPHICS TERMINAL (Y/N) (DEFAULT - N)  N

DO YOU WISH TO PRODUCE A META FILE (DEFAULT - Y)  N
```

If the user answers "yes" to this question a meta file will be produced. The name of this file will be determined by the version of DISSPLA being used—check with the system manager. A full explanation of meta files can be found in the DISSPLA manual. Briefly, when the user creates a meta file, plot information is sent to the file instead of the screen. A variety of other programs can then read this file and plot it on a screen or make hard copies.

7.1.4 File Data

The first question in this section determines if plot control information is to be read from a previously generated response file. The responses to all EXPLOT questions are recorded on the response file. This file can be saved, edited and used in a future EXPLOT session. Any system text editor can be used to edit these files.

Question (2) asks for user definition of the SAVE file(s) that contain the desired plot data. This continues until a return is entered for question 3 or until the limit of 20 files is reached. Multiple SAVE files can be used simultaneously. Questions in the graph data section define which SAVE file is to be used for a particular curve on a graph.

Question (4) asks the name of the file where the user's commands are to be stored. Answers are always saved and the user must provide an answer to this question.

Question (5) asks if the brief mode is desired. If the answer to this question is "yes" the user will be asked for a file name. This file will be a list of user defined defaults. If this file does not exist (i.e., a return was entered) the built-in defaults are used.

Example from a plot session:

```
ENTERING INTO FILE DATA SECTION

1)  DO YOU HAVE A PLOT COMMAND FILE TO BE READ  (Y/N) (DEFAULT-N)  [RET]

2)  ENTER 1ST SAVE FILE NAME OR RETURN  C360.TSV

3)  ENTER 2ND SAVE FILE NAME OR RETURN  [RET]

4)  ENTER OUTPUT FILE NAME  OUTPUT

5)  DO YOU WISH TO USE THE BRIEF MODE (DEFAULT - NO)  [RET]
```

7.1.5 Page Data

This section is used to enter the page information which consist of the page title, the page dimensions, the number of plots on a page and the number of pages to plot. The maximum number of plots on a page is 3.

The page dimension is entered as 11X8.5, 8.5X11, 10.5X10.5 etc. The default (11X8.5) is a standard page sideways (landscape mode). This default allows the largest single graph on a page.

Page data example:

```
ENTERING PAGE DATA

ENTER PAGE TITLE (DEFAULT IS BLANKS)  SAMPLE PAGE TITLE

ENTER PAGE DIMENSION (DEFAULT  (WIDTH X HEIGHT) - 11X8.5)  [RET]

ENTER # OF GRAPHS ON PAGE (DEFAULT - 1)  [RET]

ENTER # OF PAGES TO PLOT (DEFAULT - 1)  [RET]
```

7.1.6 Graph Data

This data is entered for each plot on the page(s). The questions are meant to be self explanatory.

The user may have up to 3 numbered y-axes. If more than one axis is chosen the user will be prompted for a title for each one. Further, the parameters to be plotted must be grouped by axis: all the parameters for the first axis will be input first, then the second axis, etc. After every input the program will provide an opportunity to go on to the next axis.

If more than one y axis is chosen, automatic scaling must be used. It is recommended that automatic scaling always be used. If it is not, the program will ask for the plot xstep, ystep, xmin, ymin, xmax and ymax. See the DISSPLA manual for an explanation of these terms. The word AUTO can be entered in response to any of these questions.

Parameter entry:

configuration name ..... from a BUILD card.

submodel name ..... from a BUILD card.

parameter ..... T100, G100, TL13, UA33, FR92, etc.

Valid parameters include all translatable network element parameters, but exclude control constants and user constants.

If more than one SAVE file is being used the user will also be asked for the file name associated with the parameter.

The user may turn grid lines on or off, have a box axis or an axis crossed in the center of the plot area, and may place the legend (if any) in any corner of the plot.

The last graph data entries allow the user to input a function for the x and/or y values. This feature has a number of uses. If one SAVE file is in seconds and one is in hours, the x values of one SAVE file can be converted. Data that occurred at separate times can be shown as concurrent. The functions can be used however the user wishes and are done on a per SAVE file per plot basis. The function can have any form that consists of +, -, /, * and no parentheses. If such arithmetic operations are used, note that they are evaluated from right to left *without* any precedence such as that used in Fortran.

Examples of functions:

X*3600.0 or **X*3600.0+2.0** or **X*3600.0/X+2.0** etc.

Y+460.0 or **Y*1.8+32.0** or **Y*1.8/Y+32.0** etc.

Graph data example:

```
ENTERING GRAPH DATA

ENTER GRAPH TITLE (DEFAULT IS BLANKS) SAMPLE GRAPH TITLE

ENTER X AXIS LABEL (DEFAULT TIME - HOURS) [RET]

ENTER NUMBER OF Y AXES (DEFAULT - 1) [RET]

ENTER 1ST Y AXIS LABEL (DEFAULT - TEMPERATURE DEG F) [RET]

DO YOU WISH TO HAVE AUTO SCALING (Y/N) (DEFAULT - YES) [RET]

ENTER # OF CURVES TO PLOT (DEFAULT - 1) [RET]

CURRENT CONFIGURATION NAME IS ALL
ENTER NEW CONFIGURATION NAME OR RETURN CD360

CURRENT SUB-MODEL NAME IS NONE
ENTER NEW SUB-MODEL NAME OR RETURN MAIN

ENTER PARAMETER TO BE PLOTTED (EX. T100) T111

ENTER CURVE LEGEND (DEFAULT IS BLANKS) SAMPLE LEGEND

ENTER CORNER FOR LEGEND (LL,UL,LR,UR DEFAULT - LL) [RET]

DO YOU WISH TO HAVE AUTO SCALING (Y/N) (DEFAULT - YES) [RET]

ENTER # OF CURVES TO PLOT (DEFAULT - 1) [RET]

CURRENT CONFIGURATION NAME IS ALL
ENTER NEW CONFIGURATION NAME OR RETURN CD360

CURRENT SUB-MODEL NAME IS NONE
ENTER NEW SUB-MODEL NAME OR RETURN MAIN

ENTER PARAMETER TO BE PLOTTED (EX. T100) T111

ENTER CURVE LEGEND (DEFAULT IS BLANKS) SAMPLE LEGEND

ENTER CORNER FOR LEGEND (LL,UL,LR,UR DEFAULT - LL) [RET]

DO YOU WISH TO SHOW GRID LINES (Y/N) (DEFAULT - YES) [RET]

DO YOU WISH A CROSSED AXIS (Y/N) (DEFAULT - NO) [RET]

DO YOU WISH AN X FUNCTION FOR SAVE FILE C360.TSV (Y/N) (DEFAULT - NO) [RET]

DO YOU WISH A Y FUNCTION FOR SAVE FILE C360.TSV (Y/N) (DEFAULT - NO) [RET]

DO YOU WISH TO ENTER MORE PLOTS? (Y/N) Y

ENTERING PAGE DATA

ENTER PAGE TITLE (DEFAULT IS BLANKS) PAGE TITLE 2

ENTER PAGE DIMENSION (DEFAULT (WIDTH X HEIGHT) - 11X8.5) [RET]

ENTER # OF GRAPHS ON PAGE (DEFAULT - 1) [RET]
```

```
ENTER # OF PAGES TO PLOT (DEFAULT - 1)  [RET]

ENTERING GRAPH DATA

ENTER GRAPH TITLE (DEFAULT IS BLANKS)  GRAPH TITLE 2

ENTER X AXIS LABEL (DEFAULT TIME - HOURS)  [RET]

ENTER NUMBER OF Y AXES (DEFAULT - 1)  [RET]

ENTER 1ST Y AXIS LABEL (DEFAULT - TEMPERATURE DEG F)  [RET]

DO YOU WISH TO HAVE AUTO SCALING (Y/N) (DEFAULT - YES)  N

ENTER XSTEP VALUE (DEFAULT - AUTO)  [RET]

ENTER YSTEP VALUE (DEFAULT - AUTO)  [RET]

ENTER XMIN VALUE (DEFAULT - AUTO)  [RET]

ENTER YMIN VALUE (DEFAULT - AUTO)  [RET]

ENTER XMAX VALUE (DEFAULT - AUTO)  [RET]

ENTER YMAX VALUE (DEFAULT - AUTO)  [RET]

ENTER # OF CURVES TO PLOT (DEFAULT - 1)  [RET]

CURRENT CONFIGURATION NAME IS CD360
ENTER NEW CONFIGURATION NAME OR RETURN  [RET]

CURRENT SUB-MODEL NAME IS MAIN
ENTER NEW SUB-MODEL NAME OR RETURN  [RET]

ENTER PARAMETER TO BE PLOTTED (EX.t100)  T111

ENTER CURVE LEGEND (DEFAULT IS BLANKS)  [RET]

DO YOU WISH TO SHOW GRID LINES (Y/N) (DEFAULT - YES)  [RET]

DO YOU WISH A CROSSED AXIS (Y/N) (DEFAULT - NO)  [RET]

DO YOU WISH AN X FUNCTION FOR SAVE FILE C360.TSV (Y/N) (DEFAULT - NO)  Y

ENTER FUNCTION  X*3600.0

DO YOU WISH A Y FUNCTION FOR SAVE FILE C360.TSV (Y/N) (DEFAULT - NO)  Y

ENTER FUNCTION  Y+460.0

DO YOU WISH A Y FUNCTION FOR SAVE FILE C360.TSV (Y/N) (DEFAULT - NO)  [RET]

DO YOU WISH TO ENTER MORE PLOTS?  (Y/N)  N
```

7.1.7 The Brief Mode

The brief mode is a means to eliminate (in the interactive mode only) a number of questions which are primarily concerned with page format. The brief mode can always be used, but is most suited for those situations where a large number of plots are to be created that all use the same defaults. The following paragraphs demonstrate the brief mode input.

In the file section a question was asked about the brief mode:

```
DO YOU WISH TO USE THE BRIEF MODE (DEFAULT - NO)
```

If the answer to this question was "yes" then it would be followed by the question:

```
ENTER NAME OF DEFAULT VALUES FILE (DEFAULT - NONE)
```

If a return (NONE) was entered in response to this question, then the built-in program defaults would be used, as presented in Table 7–1.

The brief mode can only be used in an interactive EXPLOT session. The brief mode can be used with or without a default input file. Without an input file the built-in program defaults are used, which are acceptable for most situations. For those situations where the defaults are not acceptable, an input file containing the default information must be created using a system editor prior to the EXPLOT session. The name of this file is the one entered in response to the question:

```
ENTER NAME OR DEFAULT VALUES FILE (DEFAULT - NONE)
```

The format of this file is presented below. This format is the same as the corresponding parts of the response file. The only column dependence in this file is that the headers PAGEDAT and GRAPHDAT must start in column 1.

Table 7–1 EXPLOT Default Values

Page Data	Default Value	Variables Affected
Page Title	blanks	TITLPG
Page Dimensions	11X8.5	XYDIMEN
Graph per page	1	NUMGR
Number of pages	1	NUMPAG

Graph Data	Default Value	Variables Affected
X axis label	TIME - HOURS	XLABEL
Number of Y axis	1	NUMAXI
Y axis label	TEMPERATURE DEG F	YLABEL
Auto Scaling	YES	XSTEP, YSTEP, XMIN, YMIN, XMAX, YMAX
Grid Lines	YES	DOGRID
Crossed Axis	NO	GRAFX
X&Y functions	none	XFUNC, YFUNC

Brief Input file example:

```
PAGEDAT
         TITLPG          =    PAGE TITLE 2
         XYDIMEN         =    11X8.5
         NUMGR           =     1
         NUMPAG          =     1
GRAPHDAT
         TITLGR          =    GRAPH TITLE 2
         XLABEL          =    Time-hours
         NUMAXI          =     1
         YLABEL          =    Temperature  Deg. F
         XSTEP           =    AUTO
         YSTEP           =    AUTO
         XMIN            =    0.0
         YMIN            =    AUTO
         XMAX            =    10.0
         YMAX            =    AUTO
         DOGRID          =    YES
         GRAFX           =    NO
         XFUNC
          1 X*3600.0
         YFUNC
          1
```

Although this file has the format of part of the response file, it is in no way a substitute for nor does it relate to the response file. The brief mode is only a means of reducing the required input information in an interactive session.

As in the command file, there must be one YLABEL entered for each axis entered for NUMAXI, and there must be a function line for each SAVE file:

```
NUMAXI          =     2
YLABEL          =    Temperature Deg. F
YLABEL          =    Temperature Deg. C
XFUNC
 1 X*3600.0
 2
 3
YFUNC
 1
 2 Y+460.0
 3
```

If the user choses the brief mode he will not be asked to answer those questions he has already answered in the input file. He will only be asked those questions listed below:

1. Graph Title
2. Number of Curves to plot
3. Configuration Name
4. Submodel Name
5. Parameter
6. Legend
7. Corner for Legend

These questions will be asked for each graph. They will be asked regardless of the manner in which the default information was specified.

7.1.8 Miscellaneous Information

There are a number of special characters available to the user to access special alphabets and characters. See the DISSPLA manual for more information on this subject.

[..... STANDARD
] ..... L/CSTD
! ..... SPECIAL
..... MATHSPECIAL
{ ..... INSTRUCTION
} ..... L/CGREEK

Special character example:

```
YLABEL          = Temperature  EH.5[0 EXHX[F
```

This string produces: Temperature °F. This is what will actually appear if the default TEMPERATURE DEG F is chosen.

7.1.9 Sample Command File

A complete command file is always output from EXPLOT sessions. For batch plotting sessions a command file must be used. The command file must have the sections outlined below. However, the user need not input all of the information shown in each section. The GLOBAL section must be complete. In all other sections, the user need only input essential data. Defaults will be used for those questions which are not answered. An examination of the input required in the brief mode will show which inputs are essential.

The command file can be prepared or edited by the user and is also output from an EXPLOT session. This file has several sections. The GLOBAL section only appears once and is always the first section of a command file. This section contains the global file information.

The next section is the PAGEDAT section. This section appears once per "page". In this section the user will be asked for general page format information as well as the number of pages and the number of graphs per page. One "page" is the number of graphs times the number of pages.

A PAGEDAT section is followed by the appropriate number of GRAPHDAT sections. there is one GRAPHDAT section for each graph indicated in the PAGEDAT section. For example, if there were 3 pages and 3 graphs per page input to the PAGEDAT section, the PAGEDAT section would be followed by 9 GRAPHDAT sections. GRAPHDAT contains the graph format information as well as the information on what to plot.

The format of this file is self evident. The main column dependence is the headers (GLOBAL,PAGEDAT,GRAPHDAT) which must start in column one.

The format of the Y axis label is presented below. The integer NUMAXI specifies the number of Y axis on the plot (maximum of 3). This is followed by the appropriate number of labels (one label per axis).

```
NUMAXI          = 2
YLABEL          = Temperature Deg. F
YLABEL          = Temperature Deg. C
```

The format of the curve data is presented below. The string CURVE DATA indicates the start of curve data. The following line contains the SAVE file number, configuration name, submodel name, parameter type, parameter actual number, and Y axis number (1-3). The next line is the legend string. This string starts in column 1. If no legend is desired this should be a blank line. This two line combination of data and legend repeats until data for all the curves to be plotted has been input. Up to 50 curves may appear on one graph.

```
        CURVE DATA
        1 CD360         MAIN      T    111  1
SAMPLE LEGEND 1
        1 CD360         MAIN      T    114  1
SAMPLE LEGEND 2
```

The SAVE file number corresponds to the information entered into file data. The first SAVE file entered is 1, the third is 3, etc.

The format of X and Y function data is started by the string X(Y)FUNC. The next line is a SAVE file number followed by the function. If no function is desired this should be just a SAVE file number. There must be one function line per SAVE file.

Sample command file:

```
GLOBAL
    TSAVE          = C360.TSV
    OUTPUT         = OUTPUT
PAGEDAT
    TITLPG         = SAMPLE PAGE TITLE
    XYDIMEN        = 11X8.5
    NUMGR          =  1
    NUMPAG         =  1
GRAPHDAT
    TITLGR         = SAMPLE GRAPH TITLE
    XLABEL         = Time-hours
    NUMAXI         =  1
    YLABEL         = Temperature  EH.5[0 EXHX[F
    NUMCRV         =  1
    XSTEP          = AUTO
    YSTEP          = AUTO
    XMIN           = AUTO
    YMIN           = AUTO
    XMAX           = AUTO
    YMAX           = AUTO
    LEGCOR         = LL
    CURVE DATA
     1 CD360       MAIN      T          111        1
SAMPLE LEGEND
    DOGRID         = YES
    GRAFX          = NO
    XFUNC
     1
    YFUNC
     1
PAGEDAT
    TITLPG         = PAGE TITLE 2
    XYDIMEN        = 11X8.5
    NUMGR          =  1
    NUMPAG         =  1
GRAPHDAT
    TITLGR         = GRAPH TITLE 2
    XLABEL         = Time-hours
    NUMAXI         =  1
    YLABEL         = Temperature  EH.5[0 EXHX[F
    NUMCRV         =  1
    XSTEP          = AUTO
    YSTEP          = AUTO
    XMIN           = AUTO
    YMIN           = AUTO
    XMAX           = AUTO
    YMAX           = AUTO
    LEGCOR         = LL
    CURVE DATA
     1 CD360       MAIN      T          111        1
    DOGRID                     = YES
    GRAFX                      = NO
    XFUNC
     1 X*3600.0
    YFUNC
     1
```

7.2 DATA_ONLY: ASCII Tabulations of SAVE Files

DATA_ONLY is a subset of EXPLOT intended for those users on machines other than a VAX, or for those who lack DISSPLA or who have alternate plotting packages available, perhaps on machines other than the platform hosting SINDA/FLUINT. A common use of DATA_ONLY is to produce plotting data for import into plotting packages on personal computers such as IBM PCs or Macintoshes.

However, because it extracts and tabulates data from SAVE files, the uses of DATA_ONLY are more general—it can be used to peruse a SAVE or RSO file, and can be used as an alternate output capability by calling SAVE (or RESAVE) within OUTPUT CALLS (provided sufficient data storage is available). When used as an output capability, DATA_ONLY is unique in allowing the user to decide which data warrants output *after* a run has been completed.

DATA_ONLY is strictly interactive, and uses the same prompts as does EXPLOT, with the same meanings and the same input requirements (refer to the previous section). DATA_ONLY does not produce a response file, and does not accept command or brief files.

DATA_ONLY produces the two output files mentioned in the previous section: EXPLOT.MIN_MAX and EXPLOT.DATA_POINTS. The file EXPLOT.MIN_MAX contains the maximum and minimum data values for each tabulated parameter along with annotation identifying the parameter. EXPLOT.DATA_POINTS contains a tabulation of each parameter along with the corresponding time values and other data identifying the parameter. Users can easily write a program to read these standard files and produce a reformatted file for their selected plotting package, spreadsheet software, etc.

7.3 RAPPR: Speed Improvements Using Simplified Fluids

In most FLUINT runs, the portion of the code that absorbs the most CPU time is the property library. The processor execution may be accelerated by substituting a fast but limited description of a working fluid: by substituting a 6000, 8000, or 9000 series fluid description for one of the 20 standard library descriptions. RAPPR (Rapid Properties) creates such reusable, custom FPROP DATA blocks. The use of a RAPPR description in lieu of a full library description results in savings on the order of 50% for a two-phase model (6000 fluid), 30% for an all-gas model (8000 fluid), and 85% for an all-liquid model (9000 fluid). Refer to Section 3.13 for more details on FPROP DATA blocks.

The trade-off is that the RAPPR-generated description is valid over a narrower range than the full library description, and involves more approximations. The user can influence the valid domain by accepting a larger error relative to full library description*. Another nuance is that FLUINT is not always capable of recovering when certain models hit the property range limits; because they have smaller limits, these simplified descriptions may encounter this problem more often. Fortunately, this condition is rare—the user is strongly encouraged to use a RAPPR description where possible.

Alternatives to RAPPR exist for models requiring two–phase or non–ideal gas descriptions of water and ammonia. Tabular descriptions are available for these fluids in the form of 6000 series FPROP DATA blocks. These tabular descriptions execute as fast as a RAPPR–generated 6000 series description, but are preferable because they are more accurate over a much wider range of pressures. Similar descriptions are available for fluids not currently found within the standard library, including hydrogen, nitrogen, and oxygen.

7.3.1 RAPPR Usage

RAPPR is designed to accommodate a typical analysis scenario. In the course of debugging and fine tuning a model, the user often makes many repeat runs. Only the final run really requires accurate properties. Fluid submodels rarely exercise more than a fraction of the total available domain. RAPPR exploits these two facts, providing the user with a mechanism for simplifying and speeding up the preliminary analyses. The final analysis can return to the full library description if desired by simply altering the fluid identified on the HEADER FLOW DATA record. Although FPROP DATA blocks may be reused repeatedly in any model, the user may rerun RAPPR to create a more appropriate fluid description (e.g., altered range limits or reference point, as presented below) at any time. As usual the INCLUDE option is recommended when using FPROP DATA blocks, such that the input file need not be changed if a new fluid description is required.

RAPPR is a once-through external program that generates a single FPROP DATA block with each execution. The user is welcome to customize a RAPPR-generated FPROP block before using it in a FLUINT analysis. Most of RAPPR required inputs are self-explanatory, and most are dependent on the type of fluid identified.

---

* The reader should know that the standard description is at best 3% accurate, at worst 10%, and that various references will disagree by at least that margin.

The user may use either set of units (SI or English). The selected unit set will be used both for RAPPR inputs and for the generated FPROP DATA block. *The choice of units is arbitrary; the generated FPROP block may be used in any model—unit conversions are handled automatically.* Within the generated FPROP DATA block, both temperatures and pressures are generated in absolute units for simplicity.

The user must input the maximum allowable error between the generated FPROP DATA block properties and the "exact" library descriptions. The range limits are calculated to maintain that criterion. Hence, the greater the tolerated error, the greater the allowable range limits that will be produced in the generated FPROP DATA block.

The user is asked to name the ASHRAE identifier for one of the 20 standard library fluids, and to name the four digit integer identifier that will be used in the generated FPROP block. While the last three digits are arbitrary, the first digit (6, 8, or 9) signals both to RAPPR and to FLUINT the type of fluid to be used, as described below.

To aid self-documentation of the resulting FPROP blocks, RAPPR includes comments describing the user inputs (original fluid ID, error tolerances) and other information such as derived (implicit) property range limits. Samples of each type of block are included in the following sections.

7.3.2 6000 Series Inputs and Results

In general, 6000 series fluids have the widest allowable range of any fluid. However, as created by RAPPR, the 6000 series fluids are designed to be fast representations of a fluid *within or near* the saturation dome in the neighborhood of a given saturation temperature/pressure. The resulting single-phase properties contain the most error, as will be shown below. The user must supply the allowable error for both liquid and vapor properties.

A sample output is listed at the end of this subsection. The keys to the speed enhancement are a one-to-one (invertible) relationship for the saturation pressure as a function of temperature, and a simplistic P-v-T surface: a relationship similar to the perfect gas formulation. A more complicated P-v-T relationship not only slows down the specific volume (VSV) calculation, but also means that much more complicated energy relationships are required for consistency. Note that the C_p for vapor is explicitly provided, but that the C_p for liquid (as well as other liquid energy properties) will be calculated by FLUINT to be consistent with other inputs. Similarly, FLUINT could have been used to calculate other implied functions such as VTS, VDPDTT, VDPDT, and VTAV2. However, because of the simplicity of the underlying relationships, these functions were input explicitly to avoid unnecessary internal iterations.

The keys to accuracy near the desired temperature (300K in the sample case) are:

1) Temperature varying properties are expressed as linear functions "exact" at 300K, with temperature derivatives evaluated at 300K.

2) The "perfect gas" relationship uses an artificial gas constant such that the specific volume of saturated vapor at 300K is "exact."

3) The coefficients in the VPS relationship are calculated to give "exact" values of pressure, heat of vaporization, and liquid specific heat at 300K.

Old values of calculation-intensive parameters are saved whenever possible to avoid recalculating them, saving 20 to 30% of the total model CPU time. Internal FLUINT routines HINTS and TLOOKS are exploited to reduce inputs. Explanations of these two functions and their arguments are beyond the scope of this text.

Extensive experience has been gained with this type of description, the first of which was produced manually for ammonia at 300K. This experience has shown that the errors in single-phase properties do not have very much effect in most applications: they are not as bad as they might appear. Typically, the two properties with the most error (e.g., the limiting factors that usually define the valid domain) are liquid C_p, and vapor specific volume. For various reasons, errors in these properties of up to 20% (as shown in this example) will make very little difference in the final results of most analyses.

The following description is a RAPPR-generated 6000 series description of ammonia at a saturation temperature of 300K. The maximum liquid (saturation) temperature is printed in the comments. This derived value is implied by the RAPPR-calculated value of PGMAX.

Sample Output for Two–Phase Fluid:

```
HEADER FPROP DATA, 6717, SI , 0.0
C
C   THIS FLUID DESCRIPTION WAS CREATED BY RAPPR FOR FLUID  717
C   AND IS ACCURATE FOR TEMPERATURES NEAR   300.00
C   MAXIMUM ERROR IN LIQUID PROPERTIES = 20.00 PERCENT
C   MAXIMUM ERROR IN VAPOR PROPERTIES  = 20.00 PERCENT
C   MAXIMUM LIQUID (SATURATION) TEMP. =   342.44751
C
         TCRIT =     405.59998    ,    PCRIT =     11282378.
         TGMAX =     361.68341    ,    PGMAX =     3209857.5
         TMIN =      264.66101
C
VINIT
       DOUBLE PRECISION POLD,PSAT
       SAVE TOLD,TTOC,A,B,BINV,C,CINV,TREF,DLIQ,DLDT
       SAVE CPVAP,CPVDT,REFF,BC,POLD,TTOCX,TSAT,DLDP,DLDPT
       DATA A,B,BINV/  20.745588    , -13366.128    ,-7.48159800E-05/
       DATA C,CINV,TREF/ -1.3277444    ,-0.75315702    ,  300.00000    /
       DATA DLIQ,DLDT/  600.23163    , -1.5193480    /
       DATA CPVAP,CPVDT,TOLD,TSAT/  3010.7249    ,  23.571568    ,2*0.0/
       DATA REFF,BC,POLD/  429.15460    ,  17746.801    ,0.0D0/
       DATA DLDP,DLDPT/ 9.38403730E-07, 1.22651453E-08/
C
VSV
       V      = REFF*T/SNGL(P)
C
VDL
       D      = DLIQ + (T-TREF)*DLDT
C
VCPV
       CP     = CPVAP + (T-TREF)*CPVDT
C
VTAV2
       T      = TLOOKS(H,CPVAP,CPVDT,TREF,0.0,0.0)
       V      = REFF*T/SNGL(P)
C
```

```
      VH
            H       = HINTS(CPVAP,CPVDT,TREF,0.0,T)
      C
      VTS
            IF(P .NE. POLD)THEN
                 TTOCX   = (ALOG(SNGL(P))-A)*BINV
                 TSAT    = TTOCX**CINV
                 POLD    = P
            ENDIF
            T       = TSAT
      C
      VDPDT
            IF(P .NE. POLD)THEN
                 TTOCX   = (ALOG(SNGL(P))-A)*BINV
                 TSAT    = TTOCX**CINV
                 POLD    = P
            ENDIF
            DPDT    = BC*SNGL(P)*TTOCX/TSAT
      C
      VHFG
            IF(T .NE. TOLD)THEN
                 IF(T .EQ. TSAT)THEN
                      TTOC    = TTOCX
                      TOLD    = TSAT
                 ELSE
                      TTOC    = T**C
                      TOLD    = T
                 ENDIF
            ENDIF
            VLIQ    = 1.0/(DLIQ + (T-TREF)*DLDT)
            HFG     = BC*SNGL(P)*(V-VLIQ)*TTOC
      C
      VPS
            IF(T .NE. TOLD)THEN
                 IF(T .EQ. TSAT)THEN
                      TTOC    = TTOCX
                      TOLD    = TSAT
                 ELSE
                      TTOC    = T**C
                      TOLD    = T
                 ENDIF
            ENDIF
            P       = DBLE(EXP(A+B*TTOC))
      C
      VDPDTT
            IF(T .NE. TOLD)THEN
                 IF(T .EQ. TSAT)THEN
                      TTOC    = TTOCX
                      TOLD    = TSAT
                 ELSE
                      TTOC    = T**C
                      TOLD    = T
                 ENDIF
            ENDIF
            DPDT    = BC*EXP(A+B*TTOC)*TTOC/T
      C
      VCONDF
            COND    =  0.47709125      + (T-TREF)*(-2.32005352E-03)
      C
      VVISCF
            VISC    =  1.40722201E-04 + (T-TREF)*(-1.53910446E-06)
```

```
C
VCONDV
      COND   =  2.46447250E-02 + (T-TREF)*( 1.05218322E-04)
C
VVISCV
      VISC   =  1.02532385E-05 + (T-TREF)*( 3.60242893E-08)
C
VST
      ST     =  1.96885522E-02 + (T-TREF)*(-2.33978993E-04)
C
VDLC
      IF(T .NE. TOLD)THEN
           IF(T .EQ. TSAT)THEN
                TTOC   = TTOCX
                TOLD   = TSAT
           ELSE
                TTOC   = T**C
                TOLD   = T
           ENDIF
      ENDIF
      PSAT   = DBLE(EXP(A+B*TTOC))
      DSAT   =   600.23163     + (T-TREF)*( -1.5193480    )
      DLD    =  9.38403730E-07 + (T-TREF)*( 1.22651453E-08)
      D      = DSAT + MAX(0.0,SNGL(P-PSAT))*DLD
C
VS
      PHI    = SINTS(CPVAP,CPVDT,TREF,  264.66101    ,T)
      S      = PHI - REFF*ALOG(REFF*T/(V*  308877.78    ))
```

7.3.3 8000 and 9000 Series Inputs and Results

8000 series fluids are "perfect" gases, and can never condense. 9000 series fluids are incompressible liquids, and can never boil. The user may choose one of two basic options for RAPPR-generated 8000 and 9000 series fluids: reference-style or array-style. While normally these styles can be mixed within such a block, when using RAPPR the user must choose between them. Of course, the user could run RAPPR twice and then use an editor to combine the two resulting FPROP DATA blocks, but the range limits should then be selected to yield the smallest domain to preserve accuracy.

For 8000 series fluids, the vapor specific volume is always "exact" at the reference temperature and pressure, and a pseudo-perfect gas constant is input to accommodate variations in temperature and pressure. Other properties are created in either array- or reference-style. The user must decide whether vapor properties are to be calculated along a line of constant pressure (isobar) or along the saturation curve. Because the standard library description does not include the effect of pressure on vapor transport properties (conductivity and viscosity), the difference between these two options affects only the range limits and the C_p values. If the isobar option is chosen, the user must input an absolute pressure in the appropriate units.

7.3.3.1 Reference-Style Format

Reference-style means the generated FPROP DATA block will use a simple linear approximation for a property in point-slope form, where all properties are "exact" at the input reference temperature, and their derivatives with temperature are also included to account for first-order variations. Temperature range limits are calculated and specified in the FPROP block to preserve the user-specified accuracy. Conductivities are typically well approximated by this style over the entire domain.

The following description is a reference-style gas-only description of refrigerant 11 at 25°C and one atmosphere, taken along an isobar. The lower temperature limit corresponds to the saturation dome.

Sample Output for Single-phase Reference-style Fluid:

```
HEADER FPROP DATA, 8011, SI , 0.0
C
C   THIS FLUID DESCRIPTION WAS CREATED BY RAPPR FOR FLUID   11
C   AND IS ACCURATE FOR TEMPERATURES NEAR   298.15
C   MAXIMUM ERROR IN VAPOR PROPERTIES  = 10.00 PERCENT
C   PROPERTIES TAKEN ALONG ISOBAR, PRESS. =   1.01325E+05
C
        TMIN =      296.96677    ,    TMAX =      612.66333
        RGAS =      58.210796    ,    TREF =      298.14999
        CP =        588.68433    ,    CPTC =     0.62976760
        K =       7.83221610E-03,    KTC =      4.10964203E-05
        V =       1.09027615E-05,    VTC =      3.05949968E-08
```

If saturation properties had been used instead, the following FPROP block would result:

```
HEADER FPROP DATA, 8011, SI , 0.0
C
C   THIS FLUID DESCRIPTION WAS CREATED BY RAPPR FOR FLUID   11
C   AND IS ACCURATE FOR TEMPERATURES NEAR   298.15
C   MAXIMUM ERROR IN VAPOR PROPERTIES  = 10.00 PERCENT
C   PROPERTIES ARE FOR SATURATED VAPOR
C
        TMIN =      209.53400    ,    TMAX =      363.19955
        RGAS =      58.108986    ,    TREF =      298.14999
        CP =        589.69684    ,    CPTC =      1.4040257
        K =       7.83221610E-03,    KTC =     4.10972789E-05
        V =       1.09027615E-05,    VTC =     3.06174464E-08
```

7.3.3.2 Array-Style Format

Array-style inputs have a much larger domain than do reference-style inputs. In fact, for 9000 fluids the properties always cover the whole temperature range from the triple (freezing) point to the critical point (or the maximum allowed saturation temperature, it less); the reference temperature is ignored only in this special case. The price paid for this extended range is slower execution, caused mostly by the extra resolution in the C_p value. For 8000 fluids the temperature range limits are caused exclusively by the vapor specific volume.

The program builds arrays to represent the properties within the user-specified error tolerance *such that a minimum number of array points are required.* For 9000 fluids, the tighter the tolerance, the more the array points are required to achieve the desired accuracy since the range limits are invariant. For 8000 fluids, the number of array elements is approximately invariant; the tighter the tolerance, the smaller the domain and therefore the same number of array points provides greater accuracy.

The following description is an array-style liquid-only description of water. Because liquid properties are independent of pressure, these arrays describe saturated liquid. Notice that only two points completely define the conductivity to within 10%, whereas considerably more points are required to adequately describe the viscosity. As the temperature approaches the critical point, the temperatures become more closely spaced. This close spacing will not usually increase the cost of the analysis since arrays are evaluated and integrated starting at the low temperature end.

Sample Output for Single-phase Array-style Fluid:

```
HEADER FPROP DATA, 9718, ENG, 0.0
C
C   THIS FLUID DESCRIPTION WAS CREATED BY RAPPR FOR FLUID  718
C   PROPERTIES ARE ACCURATE FOR SATURATED LIQUID OVER FULL
C   RANGE WITH MAXIMUM ERROR IN LIQUID PROPERTIES  = 10.00 PERCENT
C
         TMIN =      491.70001    ,    TMAX =      1150.0000
C
AT,V,      491.70001    ,  4.3486681
           497.35831    ,  3.8916168
           503.35831    ,  3.4814467
           509.35831    ,  3.1338894
           515.35828    ,  2.8376205
           522.35828    ,  2.5449030
           529.35828    ,  2.2985866
           537.35828    ,  2.0627854
           545.35828    ,  1.8661554
           554.35828    ,  1.6822401
           564.35828    ,  1.5145209
           575.35828    ,  1.3647734
           587.35828    ,  1.2334558
           600.35828    ,  1.1200825
           615.35828    ,  1.0175258
           633.35828    , 0.92412323
           656.35828    , 0.83852232
           688.35828    , 0.76209235
           767.35828    , 0.69278020
           912.35828    , 0.77067518
           979.35828    , 0.85762501
           1037.3583    , 0.95328230
           1092.3583    ,  1.0603691
           1146.3583    ,  1.1805383
           1150.0000    ,  1.1891712
AT,K,      491.70001    , 0.33202705
           1150.0000    , 0.61937493
AT,D,      491.70001    ,  62.451759
           770.35828    ,  56.751816
           904.35828    ,  51.552200
           991.35828    ,  46.851608
           1052.3583    ,  42.567879
           1095.3583    ,  38.694172
           1125.3583    ,  35.149666
           1145.3583    ,  31.842491
           1150.0000    ,  30.815065
AT,CP,     491.70001    ,  1.1114392
           699.35828    ,  1.0213013
           931.35828    ,  1.1356506
           1002.3583    ,  1.2623291
           1049.3583    ,  1.4050903
           1084.3583    ,  1.5656128
           1111.3583    ,  1.7454224
           1131.3583    ,  1.9457397
           1144.2500    ,  2.1661987
```

Appendix A Frictional Pressure Drop Correlations

This section describes the correlations used to calculate frictional pressure drops in tubes and STUBE connectors when IPDC is not 0. See also Section 4.7.13, which describes the flow regime determination logic that is applied when IPDC=6.

IPDC specifies the number of the two-phase pressure drop correlation to use. There is only one single-phase correlation, and this correlation is sometimes used in turn by the two-phase correlations. First, the single-phase correlation will be described, followed by a short description of each two-phase correlation.

A.1 Single-phase Correlation

Single-phase pressure drops are calculated using a Darcy friction factor. This factor (as represented on a Moody chart) is a function of Reynolds number (Re) for laminar flow, and a function of both Reynolds number and wall roughness ratio (roughness over diameter, ε/D) for turbulent flow. A function from Churchill is used to analytically represent the Moody chart. This function overpredicts the friction factor somewhat in the transition region, but agrees very closely with experiments at both high and low Reynolds numbers. The formula for this function is :

$$f = 8 \times \left(\left(\frac{8}{Re}\right)^{12} + \frac{1}{(A+B)^{3/2}} \right)^{1/12}$$

$$A = \left[-2.457 \times \ln \left(\left(\frac{7}{Re}\right)^{0.9} + \frac{0.27\,\varepsilon}{D} \right) \right]^{16}$$

$$B = \left(\frac{37530}{Re}\right)^{16}$$

This simple formula was selected in favor of more complicated representations of the Moody chart because of its speed and, perhaps more importantly, its numerical smoothness. The latter is important with regard to stability during steady-state iterations or slow transients. For low Reynolds numbers, roughness has no effect and the formula reduces to the familiar $f = 64/Re$.

A.2 Two-phase Correlations

This section is prefaced by a warning to the user that there is little agreement in the community as to two-phase correlation preferences. No single correlation can produce consistently acceptable predictions for all fluids under all flow regimes. In fact, disagreements between experiment and correlation (and even between two correlations) on the order of 50% are not uncommon. The situation is even more complicated by the addition of body force or gravity variations—most of the following correlations naturally focussed on water flow under 1g conditions. FLUINT therefore offers a choice of six correlations that have been developed for several fluids.

All of the two-phase correlations for IPDC=1, 2, 3, 4 and 5 were taken from Hetsroni*. This is strongly recommended reading for users who wish to explore these options in more depth.

McAdam's Homogeneous (IPDC=1)—The simplest (and smoothest and fastest) two-phase correlations are those based on the homogeneous model, which simply means that an effective density and viscosity are used and single-phase correlations are applied. Of these, McAdam's is the most widely used. Although homogeneous models often compare poorly with other correlations and underpredict pressure drops, they are the safest generalized correlation to use for uncommon fluids, or when there is no flow regime data available, or when the analysis is complicated by rapid boiling or condensation. Also, zero gravity flow regimes are expected to be "more homogeneous" than 1g flow regimes. It is for this reason that this is the default option.

Lockhart & Martinelli, Curve Fit by Chisolm (IPDC=2)—Like most of these correlations, the Lockhart/Martinelli correlation was first developed for water. It can only be used with caution for other fluids. Also, this correlation is not acceptable in the near-critical region. Therefore, it is not recommended, but is available for use at the user's discretion. On the other hand, this correlation is still one of the best for annular flow, which is dominant in most systems.

Baroczy, Curve Fit by Chisolm (IPDC=3)—This correlation is based on empirical fits to experimental data. It is cannot be strongly recommended because it tends to oversimplify the correlating parameters and is not completely dimensionless, but is available for use at the user's discretion.

Friedel (IPDC=4)—This correlation contains an important parameter missing in previous correlations: the effects of surface tension. Also, it is based on a large experimental database covering many fluids. However, it overpredicts the pressure drop (sometimes by an order of magnitude) for the low Reynolds numbers typical of spacecraft thermal management systems.

Whalley Recommendations (IPDC=5)—This is not a correlation, but rather a quantified selection of one of the above three correlations based on (1) the relative viscosities of the phases, and (2) the order of magnitude of the mass flux. The correlation usually selected is Friedel's. Therefore, this "correlation" is recommended to users who dislike or distrust both McAdam's correlation and the following option, but don't have any alternatives.

Flow Regime Based (IPDC=6)—Refer to Section 4.7.13 for in–depth discussion, and Section 4.7.14 for descriptions of the associated slip flow modeling using twinned paths when IPDC=6. Briefly, this option causes the program to estimate the current flow regime based on void fractions, velocities, body force magnitude and direction, fluid properties, etc. Four simplified regimes are recognized: bubbly, slug, annular, and stratified. Once the regime has been determined, the pressure drop is calculated on the basis of that regime. If the paths are also twinned, interface friction and wall friction apportionment are also based on the predicted regime. *If the user dislikes the homogeneous assumption but does not have another preference, this option should be used.* The methods contained in this approach should extrapolate best to fluids other than water, and to microgravity. The only caution is that the methods used for the stratified regime do not apply well to single (homogeneous) paths. The pressure drop will be overpredicted in that case because of the use of a homogeneous void fraction, which is much higher than will occur when slip flow is modeled.

* G. Hestroni, *Handbook of Multiphase Systems*, 1982.

Appendix B Convective Heat Transfer Correlations

Whenever an HTN, HTNC, or HTNS tie (Section 4.7.5) is made between fluid and thermal submodels, FLUINT automatically invokes a set of convective heat transfer correlations for internal duct flow. These correlations all assume that the heat rate is constant over the solution interval. This heat rate is calculated based on the simple relation:

$$QTIE = AHX * H * (T - TL)$$

where:

QTIE ..... Heat rate through tie positive *into* the lump and *out of* the node
AHX ..... Heat exchange area
H ..... Heat transfer coefficient
T ..... Node temperature
TL ..... Lump temperature

The methods for calculating H are described below, according to the flow regime and the temperatures in the above equation. (With HTNS ties, the heat transfer rate is also a function of the state of the upstream lump.) Exact numeric representations of the two-phase correlations are not repeated here because of their complexity. However, the references from which they were extracted are noted. Note that all of these predictions may be high because of the absence of fouling factors.

This correlation set is called from the routine STDHTC, which is also user-callable (see Section 6.9). Note that these heat transfer predictions can be overridden by supplying subroutines with equivalent names and arguments in SUBROUTINE DATA (see Section 6.9 for subroutine descriptions). User-defined (HTU and HTUS) ties may also be used for alternate convection correlations.

B.1 Single-phase, heating or cooling

Heat transfer calculations for single-phase vapor or liquid are divided into three regimes: laminar, turbulent, and transition. The laminar flow (Re < 1960) heat transfer coefficients are given by the Nusselt number for circular isothermal ducts, namely 3.66. The Nusselt number is defined as:

$$NU = H * DH / TK$$

where:

NU ..... Nusselt number.
DH ..... Hydraulic diameter.
TK ..... Fluid thermal conductivity.

This Nusselt number (3.66) not only represents the most likely case, it is a good first guess for noncircular geometries or nonisothermal heating. If a better number is known, the user may always override these calculations.

The turbulent (Re > 6420) single-phase Nusselt number is given by the Dittus-Boelter correlation*:

$$NU = 0.023 * (RE^{**}0.8) * (PR^{**}PE)$$

where:

RE	Reynolds number.	
PR	Prandtl number.	
PE	Prandtl exponent:	0.3 for cooling, 0.4 for heating.

Transitional (1960 < Re < 6420) single-phase heat transfer is given by Hausen's correlation:**

$$NU = 0.116 * (RE^{**}(2./3.) - 125.0) * (PR^{**}PE)$$

Because the Prandtl number has no effect in laminar heat transfer, a smoothing function is used between the above correlation and the laminar Nusselt number.

As discussed in Section 4.7.5, single–phase lumps employing HTN and HTNC ties use an assumption of downstream–weighted heat transfer: the heat transfer rate is calculated on the temperature at which fluid exits the perfectly mixed lump. This implicit formulation is stable, and is entirely consistent with the fundamental assumptions of a lumped–parameter network. However, when lumps are coarsely discretized (i.e., too few lumps are used to adequately represent the flowwise temperature gradient), the heat transfer rate will be underestimated, especially if the node is perceived as representing an average wall temperature for the segment. In accordance with a lumped–parameter analysis, the user should use an adequate number of elements to capture the desired number of states when using these ties.

As an alternative, the HTNS tie is specifically designed to overcome this limitation by interpreting heat transfer as occurring over a segment defined by upstream and downstream fluid states and an average wall temperature. For steady single–phase flows, HTNS (and HTUS) ties use a log mean temperature difference (LMTD) solution that tolerates very coarse spatial resolution if the wall temperature and other properties are reasonably constant.

---

* American Society of Heating, Refrigerating, and Air Conditioning Engineers, Inc. (ASHRAE) Handbook, *1981 Fundamentals*, 1981.

** Karlekar and Desmond, *Engineering Heat Transfer*, 1977.

B.2 Two-phase Flow, Boiling

Boiling heat transfer has enjoyed a great deal of attention for many years. This attention is largely due to the fact that boiling under "constant" heat flux conditions (due perhaps to electric dissipation or nuclear heating) can result in unstable or even dangerous conditions such as high wall superheating (to the point where radiation is nonnegligible). However, FLUINT assumes a quasi-steady wall temperature, and does not perform a critical heat flux calculation (which predicts the heat flux that would cause the wall to dry up even when there is a liquid phase present). Such calculations are only available in detail for water, and have large uncertainties. Nor does FLUINT handle the case of subcooled boiling, where boiling may occur in the near-wall region even though the bulk of the fluid is subcooled (again, usually found in constant heat flux cases). Subcooled boiling conflicts with the assumptions of one-dimensional flow and thermodynamic equilibrium. Extensive boiling calculations are performed that are valid for a wide variety of fluids and flow regimes, and should be adequate for most analyses. Users are always welcome to override these calculations if they have correlations that are more tailored for individual cases.

Two fundamental boiling heat transfer regimes are recognized: nucleate and film. Nucleate boiling is characterized by the formation of bubbles at the wall that drift into the bulk liquid stream. Film boiling is characterized by vapor-dominated heat transfer. Nucleate boiling is assumed to exist for qualities from 0.0 to 0.7 only. For qualities between 0.7 and 1.0, the heat transfer coefficient is interpolated between the predictions of the nucleate correlation and the film correlation. This interpolation simulates the transition between the two regimes in a numerically smooth and stable manner that approximates the actual transition, which depends on upstream conditions and exhibits hysteresis. It should also be noted that transitions between single- and two-phase coefficients are numerically smooth with the correlations used, although heat transfer coefficients rise steeply when the liquid begins to boil.

In the limit of zero flowrate, single–phase vapor (laminar Nusselt number) heat transfer conductances are applied.

Film Boiling Correlation—The film boiling correlation is simply the single-phase Dittus-Boelter correlation for vapor using the current vapor mass fraction and void fraction. This correlation somewhat underpredicts the actual heat transfer because it neglects wall to liquid radiation and the unsteady impinging of liquid droplets on the wall. However, it provides a smooth and fast estimate and a transition between the detailed nucleate boiling predictions and the single-phase vapor heat transfer.

Nucleate Boiling Correlation—The basis of the nucleate boiling correlation is Chen's* (1963). This is a very popular correlation that is valid for most fluids although best for water. It provides a good transition from single phase liquid through the entire nucleate boiling range (which is usually further subdivided). The encodable version of this correlation was taken from Collier.**

---

* American Society of Heating, Refrigerating, and Air Conditioning Engineers, Inc. (ASHRAE) Handbook, *1981 Fundamentals*, 1981.

B.3 Two-phase Flow, Condensation

Condensation heat transfer has received much less attention that has boiling, principally because of the high heat transfer coefficients usually encountered and the thermal stability. Consequently, there have been few correlations developed that attempt to include a wide variety of fluids and a wide range of flow regimes. One such correlation (Rohsenow*) is available in the default set of heat transfer calculations. This correlation was developed for condensation in annular flows using the Martinelli parameter. Smooth transitions into the single-phase regimes are also included to avoid numerical instabilities. The transition to single-phase liquid roughly accommodates the breakdown of annular flow into slug flow at low qualities (less than about 0.1), using scaling parameters similar to the Shah correlation.

Analogous to the lack of a subcooled boiling regime, condensation is neglected unless the bulk fluid is saturated. In other words, single–phase heat transfer rates will be calculated even if the wall temperature is colder than saturation.

In the limit of zero flowrate, single–phase liquid (laminar Nusselt number) heat transfer conductances are applied.

---

* Rohsenow and Harnett, *Handbook of Engineering Heat Transfer*, 1973.

Appendix C Available Fluids and Range Limits

C.1 Standard Library Fluids

Table C–1 lists the standard library fluids that are immediately available in FLUINT. All fluids are referenced by their ASHRAE refrigerant number.

Table C–1 Available Fluids

Number	Description
11	Refrigerant 11
12	Refrigerant 12
13	Refrigerant 13
14	Refrigerant 14
21	Refrigerant 21
22	Refrigerant 22
23	Refrigerant 23
113	Refrigerant 113
114	Refrigerant 114
216	Dichlorohexafluoropropane
290	Propane
318	Octafluorocyclobutane
500	73.8% R-12, 26.2% R-152a
502	75% R-22, 25% R-12
503	40.1% R-23, 59.9% R-13
505	78% R-12, 22% R-31
506	55.1% R-31, 44.9% R-114
717	Ammonia
718	Water
1270	Propylene

The user may also describe the properties of alternate single-phase fluids such as a perfect gas (8000 series numbers), a simple incompressible liquid (9000 series), a simplified two-phase fluid (7000 series), or a complete two-phase fluid (6000 series). Alternate fluids are input using a HEADER FPROP DATA block, as described in Section 3.13. The user may also apply the RAPPR external program (Section 7.3) to produce fast, simplified FPROP DATA descriptions from the above list.

Table C–2 lists the pressure and temperature values that delimit the valid range for the standard library fluids. Excursions beyond these limits will produce and error message and FLUINT will reset the wandering lump to within limits. Note that such excursions can be disregarded within FASTIC and STDSTL runs as long as they are temporary.

Table C–2 Range Limits for Standard Library Fluids

FLUID ASHRAE NO.	LOWEST TEMP. (R)	HIGHEST LIQUID TEMP. (R)	HIGHEST VAPOR TEMP. (R)	LOWEST PRESSURE (psia)	HIGHEST VAPOR PRESSURE (psia)
11	300.00	848.07	2544.2	1.61787E-03	639.50
12	300.00	693.30	2079.9	8.99569E-02	596.90
13	271.80	543.60	1630.8	0.83375	561.30
14	204.75	409.50	1228.5	0.77399	543.16
21	300.00	800.00	2438.7	3.20782E-03	670.49
22	300.00	664.50	1993.5	0.16170	721.84
23	269.17	538.33	1615.0	0.60119	701.42
113	428.70	871.00	2631.0	0.28871	443.78
114	322.50	753.95	2261.9	2.98006E-02	473.18
216	325.00	815.67	2447.0	1.85549E-03	399.46
290	300.00	665.95	1997.9	0.23209	617.40
318	417.10	699.27	2097.8	2.5725	403.60
500	300.00	681.59	2044.8	0.12214	641.90
502	300.00	639.56	1918.7	0.26453	591.00
503	263.39	518.00	1580.3	0.76153	563.66
505	300.00	703.66	2111.0	8.80965E-02	685.59
506	300.00	747.40	2242.2	2.46063E-02	749.37
717	351.80	730.08	2190.2	0.87683	1636.4
718	491.70	1150.0	3495.3	8.76988E-02	2878.2
1270	300.00	657.09	1971.3	0.31651	667.01

FLUID ASHRAE NO.	LOWEST TEMP. (K)	HIGHEST LIQUID TEMP. (K)	HIGHEST VAPOR TEMP. (K)	LOWEST PRESSURE (Pa)	HIGHEST VAPOR PRESSURE (Pa)
11	166.67	471.15	1413.5	11.155	4.40921E+06
12	166.67	385.17	1155.5	620.23	4.11547E+06
13	151.00	302.00	906.00	5748.5	3.87001E+06
14	113.75	227.50	682.50	5336.5	3.74494E+06
21	166.67	444.44	1354.8	22.117	4.62287E+06
22	166.67	369.17	1107.5	1114.9	4.97692E+06
23	149.54	299.07	897.22	4145.0	4.83612E+06
113	238.17	483.89	1461.7	1990.6	3.05973E+06
114	179.17	418.86	1256.6	205.47	3.26249E+06
216	180.56	453.15	1359.5	12.793	2.75415E+06
290	166.67	369.97	1109.9	1600.2	4.25684E+06
318	231.72	388.48	1165.5	17737.	2.78272E+06
500	166.67	378.66	1136.0	842.14	4.42575E+06
502	166.67	355.31	1065.9	1823.9	4.07480E+06
503	146.33	287.78	877.95	5250.6	3.88631E+06
505	166.67	390.92	1172.8	607.40	4.72697E+06
506	166.67	415.22	1245.7	169.85	5.16675E+06
717	195.44	405.60	1216.8	6045.5	1.12824E+07
718	273.17	638.89	1941.8	604.86	1.98443E+07
1270	166.67	365.05	1095.2	2182.2	4.59889E+06

The rules behind the property limits are as follows, with the appropriate property function given in parentheses:

1) The lowest temperature is given by the maximum of (the freezing point and the minimum of (300R and one half of the critical temperature)). One exception: 325R for fluid 216. (VTMIN)

2) The lowest pressure is the saturation pressure corresponding to the lowest temperature. (VPS(VTMIN))

3) The highest allowable temperature for liquids is the critical temperature, or the saturation temperature for the highest allowable pressure if less than critical. (VTLMAX)

4) The highest allowable temperature for vapor is three times the critical temperature. (VTGMAX)

5) The highest allowable pressure for vapor is the critical pressure (P_c) for most fluids, or $0.9*P_c$ for fluids 21, 718, 1270, and 503. (VPS(VTLMAX))

Note that the temperature and pressure cannot be simultaneously above their critical values. The highest vapor temperature is arbitrary and is often unrealistically high, but allows a wide range for controlling excursions. The user should avoid analyses that rely on accurate properties at such high values of superheat.

For all fluids, the values of maximum and minimum conditions may be found with calls to VTMIN, VTLMAX, VTGMAX, VPS, and VPGMAX as shown above. Implicit limits (i.e., those implied by inputs but not specifically stated) of user-defined fluids are calculated by the program and echoed to the output file at the start of a run.

C.2 User Defined Fluids

Because of their simplicity, the valid ranges of user-defined single-phase fluids are large. Pressures for liquid (9000 series) may assume any value including negative values. Pressures for gases (8000 series) must be positive, but can be arbitrarily high. The valid temperature range for user-defined fluids is determined by the FPROP DATA inputs, whether explicitly or implicitly. While the temperatures must always be positive and can be arbitrarily high, the user can inadvertently limit the valid temperature range for extrapolated data. For example, viscosity must be positive. If the user input V = 1.0 and VTC = -0.01 at TREF = 300.0, then viscosity can become zero at T = 300.0+(1.0/-0.01) = 200.0 degrees. The *lower* temperature limit for that fluid would then be *at least* 200.0 degrees. While coefficient-style inputs can only cause one such limit, AT array data can be extrapolated in both directions and so cause both an upper and lower limit. FLUINT scans all data for user-defined fluids to determine the valid range at the start of the processor operation, and issues a message defining any limits and the reason for them. Afterwards, those limits are enforced and warning messages will be produced if the solution exceeds them. The user can explicitly influence temperature limits using the TMIN and TMAX input keywords.

When a two-phase working fluid is desired that is not contained in the standard library, the user may input a description of the fluid as either a 6000 series or 7000 series fluid. Section 3.13 describes implicit range limits for such fluids, and suggests explicit ones.

The range limits of a 6000 series fluid are as wide as the user sets them; this is the only fluid that can currently exceed both critical temperature and pressure. Enforcement of the limits is partially enforced by the program, but must also be enforced by the user within his functional subblocks. The range limits of a 6000 series fluid are also sensitive to the use of the NEVERLIQ option, which employs PCRIT as a lower limit and assumes that the fluid is always compressible and has a quality of 1.0.

Because of the simplified equation of state, the limits on a 7000 fluid are more restrictive. For such a fluid, the range limits may be set explicitly by the user, but the program performs consistency checks to further narrow the valid range. These checks are along the same lines as those limits noted for library fluids, except that the maximum liquid temperature and vapor pressure are usually subcritical. Furthermore, the P_{sat}-T_{sat} relationship is scanned for implicit range limits. For example, the derivative $(dP/dT)_{sat}$ must always be positive. Furthermore, the calculated specific heat for saturated liquid must always be positive.

Appendix D Unit Systems

When a fluid submodel is used in SINDA/FLUINT, the user is required to name the set of units desired using the global constant UID. There are two choices: metric (SI) and English (or U.S. Customary Engineering) units. Metric units use the MKS (meter, kilograms, seconds, Kelvin) basis. English units use feet, pounds-mass, hours, Rankine, BTUs, and psi. Table D–1 lists the assumed units for each option or parameter. Note that once a unit system is selected, all

Table D–1 Unit Systems, Assumed Units

Parameter	Metric Units	English Units
Length	m	ft
Mass	kg	lb_m
Time	s	hr
Temperature	C, K	F, R
Pressure	Pa	psia
Quality	-	-
Void Fraction	-	-
Sp. Enthalpy	J/kg	BTU/lb_m
Energy	J/kg	BTU/lb_m
Sp. Heat	J/kg–K	BTU/lb_m–R
Area	m^2	ft^2
Volume	m^3	ft^3
Density	kg/m^3	lb_m/ft^3
Flowrate	kg/s	lb_m/hr
Viscosity (Dyn.)	kg/m–s	lb_m/ft–hr
Conductivity	W/m–K	BTU/hr–ft–R
Surface Tension	N/m	lb_f/ft
Gas Constant	J/kg–K	ft–lb_f/lb_m–R
VDOT	m^3/s	ft^3/hr
QDOT	W	BTU/hr
COMP	1/Pa	1/psi
HC	Pa	psi
AC, FC*	1/m–kg	1/lb_m–ft
WRF, UPF, FPOW, FK	-	-
CFC	m^3	ft^3
GK	kg/s–Pa	lb_m/hr–psi
HK	kg/s	lb_m/hr
EI, EJ	1/J–s	1/BTU–hr
DK, AM	-	-
FD, FG	1/s	1/hr

* When FPOW, which itself is unitless, equals 1.0.
Otherwise, FC units are dependent on value of FPOW

calculations including heat transfer with thermal submodels will be performed in that system of units. The user should therefore use the same unit system throughout the entire model (FPROP blocks are a singular exception).

ABSZRO and PATMOS can be used to change the units for temperature and pressure from relative to absolute (R to F, K to C, psia to psig, etc.), and that temperature, pressure, and flowrate conversion factors (PLFACT, PLADD, TLFACT, TLADD, and FRFACT) are available to convert inputs.

Appendix E Summary of Numerical Methods

This appendix contains a summary of the mathematics behind the FLUINT solution algorithms. The derivation is too lengthy and the actual equations are too cumbersome to present here. However, the interested user will find sufficient information in this section for a general understanding of the approach and the approximations. With such an understanding, the user will be better able to apply FLUINT to unusual situations.

The basic methods described herein are used for both STDSTL and for transient analyses. For STDSTL analyses, an artificial time step is used to relax the network. These methods are incorporated into an algorithm documented in Table E–1. FASTIC uses a simple iterative relaxation method similar to solving junction equations for both tanks and junctions. See Section 6.2.5 for a discussion of FASTIC.

Nomenclature

AC ..... Tube recoverable loss (area change and acceleration) coefficient
AF ..... Tube flow area
AM ..... Tube added mass coefficient
COMP ... Tank wall compliance factor
DK ..... Path flowrate/twin's flowrate coefficient
EI ..... Path flowrate/enthalpy coefficient (defined upstream end)
EJ ..... Path flowrate/enthalpy coefficient (defined downstream end)
e ..... Flowrate rectifier (+1 if defined upstream, -1 if downstream)
FC ..... Tube irrecoverable loss (friction) coefficient
FD ..... Tube interface drag coefficient
FK ..... Tube head loss coefficient (K factor)
FG ..... Tube momentum transfer coefficient (phase generation)
FPOW ... Flowrate exponent if FC (irrecoverable loss) term
FR ..... Mass flowrate
GK ..... Connector flowrate/pressure coefficient
h ..... Donor enthalpy (includes phase suction action)
HC ..... Head coefficient (pressure, body force)
HK ..... Connector offset factor
M ..... Tank mass
PL ..... Lump pressure
Q ..... Nodal energy source or sink term
QDOT .... Lump energy source or sink term
t ..... Time
TLEN .... Tube length
U ..... Tank internal energy
VDOT .... Rate of change (growth) of tank volume
X ..... Solution vector
α ..... Area fraction (void fraction or liquid fraction)
ε ..... Phase donor rectifier (1 if recipient phase, 0 if donor phase)

Superscripts

n ..... Current time value
n+1 ..... next time value

Subscripts

k ..... path
l ..... lump
i ..... defined upstream lump
j ..... defined downstream lump
t ..... twin path

Table E-1 FLUINT Solution Algorithm in Pseudo-Code

```
Call FLOGI0.
Perform component model simulation calculations.
Perform heat transfer calculations (calculate UA, QTIE, QDOT, update junctions) [1].
Perform LINE and HX spatial acceleration (AC, FG) calculations
Perform connector device simulation calculations (calculate GK, HK, DK, EI, EJ).
Perform tube friction factor calculations (calculate FC, FPOW, FD).
Call FLOGI1.
Estimate impending changes in hard tanks.
IF pressure propagation solution is possible and necessary THEN:
        Calculate hard tank property derivatives.
        Prepare path data for network solution.
        Fill coefficient matrix and constant vector.
        If matrix structure has changed redo symbolic solution.
        Perform matrix factorization and solution.
        Update tank states and path flowrates
        Update junction solution.
        IF must refresh simulation parameters [2]THEN:
                Perform component model simulation calculations.
                Perform heat transfer calculations (update junctions).
                Perform spatial acceleration calculations for ducts
                Perform connector device simulation calculations.
                Perform tube friction factor calculations.
                Call FLOGI1.
                Estimate impending changes in hard tanks.
        END IF.
END IF.
Calculate hard tank property derivatives.
Calculate soft tank intensive property derivatives.
Find allowable time step.

[PERFORM THERMAL SUBMODEL SOLUTION.  REPEAT ABOVE ONLY FOR BACKUPS]

Calculate soft tank extensive property derivatives.
Fill coefficient matrix and constant vector.
IF matrix structure has changed THEN:
        Swap primary workspace with secondary.
        If old structure inappropriate redo symbolic solution.
END IF.
Perform matrix factorization and solution.
IF resulting solution is unacceptable THEN:
        Reset parameters and reduce time step
        Return to the start of the integration.
END IF.
Update tank states and path flowrates.
Update junction solution.
Call FLOGI2 if transient or if finished with steady-state.
Call OUTCAL if needed.
```

[1] Junction solution iterates hydrodynamic and energy equations until junction states converge.
[2] If network is all hard or if any tank has flashed, simulation parameters must be refreshed.

E.1 Basic Governing Equations

Lump Equations—Lumps are assumed to be perfectly mixed and hence are described by a single thermodynamic state. Energy and mass are conserved at every lump.

Plena are sources or sinks, supplying any necessary mass or energy needed to maintain a given thermodynamic state. There are no governing equations for plena—they appear only as terms in other equations.

Junctions have no volume and therefore no storage capacity; mass and energy flows are balanced at all times. Junctions may have a heat source/sink term, QDOT. The equations for a junction are algebraic conservation equations for mass and energy:

$$\text{Mass:} \quad \sum_k e_k \cdot FR_k = 0$$

$$\text{Energy:} \quad \sum_k e_k \cdot h_k \cdot FR_k + QDOT_l = 0$$

Tanks have a definite volume and therefore mass and energy storage capability. They have the capability to grow or shrink with time according to VDOT, and may also grow or shrink with pressure according to COMP. Tanks also have a source or sink term for heat, QDOT. The equations for tanks are therefore differential conservation equations for mass and energy:

$$\text{Mass:} \quad \sum_k e_k \cdot FR_k = \frac{dM_l}{dt}$$

$$\text{Energy:} \quad \sum_k e_k \cdot h_k \cdot FR_k + QDOT_l - PL_l \left(VDOT_l + \frac{dPL_l}{dt} \cdot VOL_l \cdot COMP_l \right) = \frac{dU_l}{dt}$$

Note that the assumption of perfect mixing means that vapor and liquid phases must be in thermal equilibrium in a two-phase tank *at all times*. This means, for example, that adding a little subcooled liquid to a saturated tank will cause an immediate pressure and temperature drop. Nonequilibrium simulations may be handled by subdividing control volumes and utilizing the NONEQ0 and NONEQ1 routines described in Section 6.9.6.

The donor enthalpy term from path k (h_k) takes into account phase suction options, twinned paths, capillary options, etc. As such, it may not be the same value for all paths flowing out of a single lump. For outflow, the donor enthalpy usually represents the enthalpy of the lump itself, but it may alternatively represent saturated liquid or saturated vapor. If the lump is a junction, the donor enthalpy may represent some state in between saturation and the upstream state, as needed to balance energy flows. The enthalpy of the lump may therefore differ from that which would be calculated from a mixing of the incoming flows because of single phase suction out of the lump.

Path Equations—Paths are assumed to be one-dimensional and hence described by a single mass flow rate. The assumption of one mass flowrate means that vapor and liquid move at the same velocity, with no phase drift or slip flow. Just as imperfect mixing analyses require more than one control volume, unequal phase velocities simulation requires more than one parallel path (as provided by the optional use of twinned paths.)

Tubes conserve momentum—the flow rate in tubes cannot change instantaneously. Rather, the fluid inside of the tube must undergo acceleration or deceleration. Hence, the governing equation for tubes is simply a complex form of Newton's second law:

$$\frac{dFR_k}{dt} = \frac{AF_k}{TLEN_k}\left(PL_{up} - PL_{down} + HC_k + FC_k \cdot FR_k \cdot |FR|_k^{FPOW_k} + AC_k \cdot FR_k^2 - \frac{FK_k \cdot FR_k \cdot |FR|_k}{2 \cdot \rho_{avg} \cdot AF_k^2}\right)$$

The exponent FPOW is a function of the current flow regime, and may vary from 0.0 (for laminar) to nearly 1.0 (for fully turbulent). Note that both the FC term and the FK term always result in a retarding force, independent of flow direction, while the AC term is a retarding force is one direction and a boost in the other. The inertia within a tube is proportional to the density times TLEN/AF (i.e., ρL/A). While the TLEN/AF appears inverted in the tube equation, the user should note that the density is contained within the FC and AC terms.

For twinned tubes, the above momentum equation becomes:

$$\frac{dFR_k}{dt} = \frac{AF_k}{TLEN_k}\left[\alpha_k (PL_{up} - PL_{down} + HC_k) + \frac{1}{\alpha_k}\left(FC_k \cdot FR_k \cdot |FR|_k^{FPOW_k} + AC_k \cdot FR_k^2 - \frac{FK_k \cdot FR_k \cdot |FR|_k}{2 \cdot \rho_k \cdot AF_k^2}\right)\right]$$

$$\frac{1}{\alpha_k} FD_k \cdot FR_k - \frac{1}{\alpha_t} FD_t \cdot FR_t + \varepsilon_k \left(\frac{1}{\alpha_k} FG_k \cdot FR_k + \frac{1}{\alpha_t} FG_t \cdot FR_t\right)$$

$$\alpha_t \cdot AM_t \cdot \frac{dFR_t}{dt} - \alpha_k \cdot AM_k \cdot \frac{dFR_k}{dt}$$

Reversing the subscripts k and t results in the equation for the second twin.

Connectors can change flow rates instantaneously. They are governed by a linear algebraic constraint equation between flow rate and pressure differential. This generic equation is a vehicle used for the simulation of a wide variety of devices, and does not represent any conservation relation. It is expressed only in differential form; that is, for changes in flowrate:

$$\text{Constraint:} \quad \Delta FR_k^{n+1} = GK_k^n (\Delta PL_i^{n+1} - \Delta PL_j^{n+1}) + HK_k^n$$

In the above equation, GK is the current partial derivative of flow rate with respect to pressure differential:

$$GK_k = \frac{\partial FR_k}{\partial (PL_i - PL_j)}$$

Note that this partial is evaluated at the flowrate that *would* exist at the current pressure differential, not at the current flowrate. This distinction may seem trivial, but it greatly increases the stability of the algorithm. HK is defined as this hypothetical flowrate minus the current flowrate:

$$HK_k = FR_k^{n+1} - FR_k^n$$

For twinned connectors, or if the user has input either EI or EJ for a NULL connector, then the full equation is:

$$\Delta FR_k^{n+1} = GK_k^n(\Delta PL_i^{n+1} - \Delta PL_j^{n+1}) + HK_k^n + EI_k^n \cdot \Delta HL_i^{n+1} + EJ_k^n \cdot \Delta HL_J^{n+1} + DK_k^n \cdot \Delta FR_t^{n+1}$$

Using more partial derivatives as defined below (also evaluated at the next hypothetical state):

$$EI_k = \frac{\partial FR_k}{\partial HL_i} \qquad EJ_k = \frac{\partial FR_k}{\partial HL_j} \qquad DK_k = \frac{\partial FR_k}{\partial FR_t}$$

E.2 Formulation of the Equation Set

The equations actually solved in FLUINT are variations of the above set. Iterative solutions of these equations are not feasible except in FASTIC—they must be solved simultaneously. This fact has three ramifications on the solution scheme: (1) the equations must be linearized, (2) the number of equations (i.e., order of the solution) must be minimized, and (3) a sparse matrix algorithm should be used. In a network consisting of L nonplenum lumps and P paths, the solution vector is 2L+P long. FLUINT reduces this to a compact set of 2L linear equations with the least restricting assumptions possible. (A smaller set of P equations could be produced, but this would disallow many program features.)

FLUINT uses differential forms of the above equations. In other words, the program solves for differences between the last solution vector and the next. The basic form of all of the finite differenced differential equations is:

$$\Delta X^{n+1} = \Delta t (\dot{X}^{n} + \Delta \dot{X}^{n+1})$$

This is a first-order implicit method that is "unconditionally stable" when $\Delta \dot{X}^{n+1}$ is known exactly. This means that an arbitrarily large time step may be taken and still produce a stable answer. However, all this formulation guarantees is that the answers won't artificially diverge—no guarantees can be made regarding the accuracy. This is especially true since the original equations were not linear. FLUINT selects an appropriate maximum time step for each method, as described in E.6, to maintain a consistent level of accuracy.

The derivative at the next time step, $\dot{X}^{n+1}$, is not known. However, the difference between this derivative and the last one, $\dot{X}^{n}$, can be estimated to a high degree of accuracy. In order to approximate this change in the derivative, it is necessary to make some assumptions. These assumptions are listed at the end of this subsection. Note that every element (e.g., lump and path) is completely described by the attributes of the adjacent elements, and not by those of more remotely located elements. Because of this, the network can be arbitrarily constructed and the above equation is comparatively simple. In mathematical terms, the Jacobian of the coefficient matrix for this system is trivial.

The algebraic equations are similarly expressed in differential form. Higher order terms in the junction energy equation must be neglected to avoid nonlinear equations. Because of this and because of computer word round-off errors, a fast iterative correction to the junction mass and energy flows is performed after each solution step.

When the tube momentum equation is finite differenced into the form described above, it can be expressed in the same form as the connector equation once the time step is known.

Thus, during each solution interval, all paths are only dependent on the changes in pressure differential and perhaps the enthalpy of endpoint lumps. This allows the final equation set to be reduced to 2L, representing an energy and a mass equation for each tank and junction. Because of the reduced size of the solution set, the mathematical expressions for the remaining equations are very lengthy and cumbersome, and so will not be presented.

The following is a list of major assumptions made within FLUINT necessary to arrive at the present formulation:

1) Lumps are perfectly mixed at all times and are at thermodynamic equilibrium (i.e., described by one state).

2) Paths have one flow rate. All fluid within the path moves at the same velocity: the flow profile is one dimensional.

3) Tube attributes (AC, FC, FPOW, HC, FD, FG, AM, FK) are constant over the solution interval. Connector attributes (GK, HK, EI, EJ, DK) are likewise constant for each interval.

4) Liquid is incompressible (infinite speed of sound). Density is a function of temperature only for liquids.

5) Specific thermodynamic derivatives (e.g., the partial of specific enthalpy with respect to density at constant pressure) for tanks are constant over each interval.

6) During the solution interval, the operation of each path is a function only of end states (lumps on each end), and the operation of each lump is a function only of the mass and energy conducted to it by adjacent lumps and of the heat and volume rate terms imposed on it. These rates (QDOT and VDOT) are also constant over the solution interval, as is COMP.

7) The second order term $\Delta h^{n+1} * \Delta FR^{n+1}$ in the junction energy equation can be neglected (if need be) during the solution, and corrected iteratively afterwards along with round-off induced mass flow rate imbalances. The first order term $h^{n+1} * \Delta FR^{n+1}$ could be included, but is also usually neglected to avoid destabilizing error terms as a result of the above assumption. In other words, junction enthalpy changes during the main solution are due to upstream enthalpy changes and not to flowrate changes. (Junctions and connectors are updated before and after the main solutions, which rectifies this oversight.)

   If twinned paths are active, their EI and EJ terms become nonzero. The summation term $\Sigma EI - \Sigma EJ$ is usually then nonzero for junctions adjacent to those paths, in which case the $h^{n+1} * \Delta FR^{n+1}$ terms in their energy equations must be reinstated along with the Δh^{n+1} term in their mass equations. Fortunately, no stability problems are caused in this special case. (It is the inclusion of such terms that forces the FASTIC matrix solution, which is normally strictly a hydrodynamics solution, to include also the energy equations—effectively doubling the vector size.)

8) Tank walls with COMP=0.0 are inflexible (infinitely stiff) even though they may grow or shrink at user prescribed rates (e.g., VDOT).

E.3 Incompressible Tanks

FLUINT assumes that the liquid state is incompressible, and therefore density is a function of temperature only. This assumption simplifies the description of the fluid and eliminates the requirement for a time step small enough to resolve liquid compressibility effects such as the propagation of pressure waves. However, since this assumption is also equivalent to an infinite speed of sound in liquid regions, special numerics must be employed to assure that accuracy requirements are met.

Tanks, therefore, are either incompressible (*hard*, even if compliant) or compressible (*soft*). They may change freely from one state to another automatically within the integration, and the user need not keep track of these transitions.

The equations for hard and soft tanks differ greatly from each other. The solution variables for soft tanks are extensive internal energy and mass, and pressure is a dependent variable. Hard tanks, on the other hand, use pressure as an independent variable, with mass a dependent variable. In soft tanks, changes cannot occur discontinuously. In hard tanks, however, pressure may change discontinuously. These facts have two ramifications on the solution methods used.

First, FLUINT may internally perform a pressure propagation solution as needed to preserve accuracy in a network with hard-filled tanks and fast changing conditions (or rough initial conditions). This solution is described in the next section.

Second, certain events may invoke a pressure propagation solution and/or suddenly decrease the integration time step. This is a result of the fact that the density in a hard tank and the pressure are unrelated, while the net mass flow rate must be always nearly balanced. The events to be aware of include sudden changes in the volume rate (VDOT) in a hard-filled tank, and, less intuitively, sudden changes in the heat rate (QDOT) and/or the enthalpy flow. This last phenomena is caused by thermal expansions or contractions, which must be accommodated by a change in net flow rate in or out of the tank.

Note that giving a tank any compliance, however small, can eliminate sudden and unrealistic pressure spikes and notches. Perhaps more importantly, pressure is a time-dependent quantity for compliant tanks, which means that the time step controller (described below) assumes responsibility for its variation. (Otherwise, the time step controller must ignore variations in time-independent quantities.)

E.4 Pressure Propagation Solution

As noted above, FLUINT may internally and itermittently perform a zero time step solution. This solution is used to instantaneously propagate pressure information throughout the hard-filled and time-independent portions of a fluid model. It is essentially the limiting case of the normal solution equation set as the time step approaches zero. Therefore, it affects only a subset of the network—those parameters that are time independent. These include the states of hard tanks with zero compliance, junction states, and connector flowrates.

Since the pressure propagation solution is a limiting case of the normal equations, the regular time-dependent full-network solution will also perform the same functions. However, there are several reasons to utilize a separate method. First, accuracy cannot be guaranteed—unfo-

reseen pressure spikes may occur that were not accounted for in the time step predictor. By taking a zero time step solution, the time step predictor will have new information to use to preserve accuracy. Second, only the hard-filled portion of the model will change, so that the solution vector for the pressure propagation solution may be very small and therefore fast.

Finally, note that FLUINT decides internally whether or not a pressure propagation solution is required. This is done by comparing the current flow rate imbalance in all of the hard-filled, noncompliant tanks and the desired accuracy (DTSIZF or RSSIZF). Too large an imbalance will cause a pressure spike since the net change in density (which is the only source of mass storage in a hard tank) is extremely small.

E.5 Phase Suction Options

FLUINT allows violations of the assumption of homogeneous flow. The user may specify that a path accepts only one phase from such a lump when that phase is present. This may be used to simulate stratified tanks, liquid/vapor separators, and is used internally with twinned paths and with capillary devices. All correlation routines obey the current phase suction status when predicting heat transfer, frictional and acceleration pressure drops, and body forces. See Section 4.7.8.

In order to accommodate such an option, a further assumption is made in these cases: the specific enthalpy of the phase being extracted is constant over the solution interval if there is enough of that phase available and the upstream lump is expected to remain in a two-phase state over the length of the next solution interval. This does not apply to junctions or FASTIC tanks for which the outflowing phase suction is incompletely satisfied, in which case the enthalpy term is allowed to vary.

E.6 Time Step Algorithm

FLUINT predicts the maximum time step allowed by each submodel to preserve the desired accuracy. This maximum time step is calculated based on both the current and previous (i.e., last time step) network status, and represents a conservative estimate.

Basically, the time step is chosen such that no key variable changes by more than DTSIZF (or RSSIZF for STDSTL) percent. These key variables include all whose derivative is known, such as tank mass, volume, extensive internal energy, and pressure (if soft). Tube flowrate changes are analogously controlled by DTTUBF (or RSTUBF).

The time step predictor must also restrict the allowable time step if a transition is imminent. These transitions include hard to soft and vice versa, the amount of time until a particular phase is depleted, anticipated deprime or reprime of capillary devices, and reversals in path flowrates. Transition time steps are calculated such that the impending transition can occur but the solution does not go very far past the discontinuity.

Section 4.7.18.2 describes the mechanics of the time step selection in more detail.

E.7 Matrix Inversion

Both the pressure propagation and the normal solution set (as well as the separated hydrodynamic solution in FASTIC) are solved simultaneously because their sensitive and highly coupled nature makes iterative solutions almost impossible. The matrix to be inverted is not symmetric but is usually very sparse.

FLUINT uses a three step sparse matrix solution: the Yale Sparse Matrix Package, which is public domain software available from the National Institute of Standards and Technology (formerly the NBS). The first step is a symbolic solution of the nonzero elements. The second and most time consuming step is an LDU factorization using actual numbers. The third step is the final production of the solution vector.

The reason that this package was selected was that not all of these steps need be repeated, saving time for similar matrices. For instance, if the coefficient matrix does not change at all, only the last step is needed after the first pass. If the elements *do* change, but the structure (location of nonzero elements) *doesn't* change, then only the last two steps need be repeated. This last example is relevant to FLUINT. If the network is changing gradually, then the coefficient matrix will have different values but the same structure, permitting the symbolic solution to be skipped. This reduces the CPU time to invert the matrix by 5 to 30%. In FLUINT, a matrix will maintain the same structure as long as no flow rates reverse direction and as long as no tanks transition from hard to soft or vice versa. Because there are many cases where a solution matrix toggles between one form and another during an integration, FLUINT saves the latest two symbolic solutions: the most recent is called the primary and the older solution is called secondary. Thus, when the matrix structure changes and the primary symbolic solution is no longer valid, FLUINT checks to see if the secondary solution is valid. If so, it becomes primary. If not, a new primary solution is generated. In either case, the former primary solution is saved as the secondary for future use.

The pressure propagation solution rarely has the same structure as the normal solution. Because of this and the fact that it is not always invoked, the pressure propagation solution always overwrites the secondary symbolic solution. Thus, if it is repeated and the primary solution maintains the same structure, only the last two inversion steps are needed. This scheme was developed to optimize both storage and CPU time requirements.

The storage of the symbolic solution requires extensive memory for large problems. The exact amount is uncertain; the preprocessor conservatively estimates the required amount. The routine YSMPWK can be called at the end of OPERATIONS DATA to report on actual usage.

Finally, it may be of interest to note that there are actually two types of inversion procedures: compacted storage and uncompacted. It was found empirically that the uncompacted method was faster and required less storage if the size of the solution vector was less than 50 (corresponding to 25 tanks and junctions) regardless of sparseness, while the compacted method was faster and required less storage for larger problems. FLUINT internally selects the optimum method.

SUBJECT INDEX

BY SECTION NUMBER

Contract
NAS9-18411
MCR-91-1393
October 1993

SINDA/FLUINT

Systems Improved Numerical Differencing Analyzer and Fluid Integrator

Version 2.6

USER'S MANUAL
APPENDIX F
Sample Problems

TABLE OF CONTENTS

SINDA

FLUINT

List of Figures

SINDA SAMPLE PROBLEM: SATELLITE DEPLOYMENT

This problem predicts the thermal transients during deployment of a cubic "satellite" from within a cubic "cargo bay." This problem was chosen to demonstrate the following SINDA features:

1. the use of submodels and BUILD commands
2. the use of steady-state and transient runs
3. the use of array data and time-varying sources (TVS)
4. the use of global user data
5. the use of user character array (CARRAY) data
6. the use of user-defined files (USER1 and USER2)

S.1 Problem Description

A hollow cubic satellite is hovering centered (i.e., negligible conductive contact) within a hollow cubic cargo bay in low earth orbit. Within the satellite a small solid cube hovers. The cargo bay is initially in a planet-oriented equatorial orbit, and is open on the side facing away from earth. When the spacecraft passes behind the earth, the satellite is deployed in the same orbit, and is reoriented to be sun-facing (see Figure S–1). The initial and final orbital-average conditions of the satellite are desired, as well as a simulation of the deployment event. The relevant characteristics are:

Cargo bay (insulated on outside):
- Number of panels 5
- Overall size ..... 2 ft x 2 ft x 2 ft
- Panel thickness ..... negligibly thin
- Material ..... Aluminum

Satellite:
- Number of panels 6
- Overall size ..... 1 ft x 1 ft x 1 ft
- Panel thickness ..... 1/8 inch
- Enclosed solid cube size ..... 1 in. x 1 in x 1 in.
- Material ..... Aluminum

All active surfaces:
- IR emissivity 0.8
- Solar absorptivity 0.2
- Insulated at edges.

S.2 A SINDA Model

The modeling choices reflect the importance of radiation exchange in the simulation and the lack of conduction between surfaces. The lack of conduction means that each surface can be characterized by a single temperature, and therefore can be adequately modeled by a single node. This assumes that the temperature drop through the thickness of the thin satellite panels

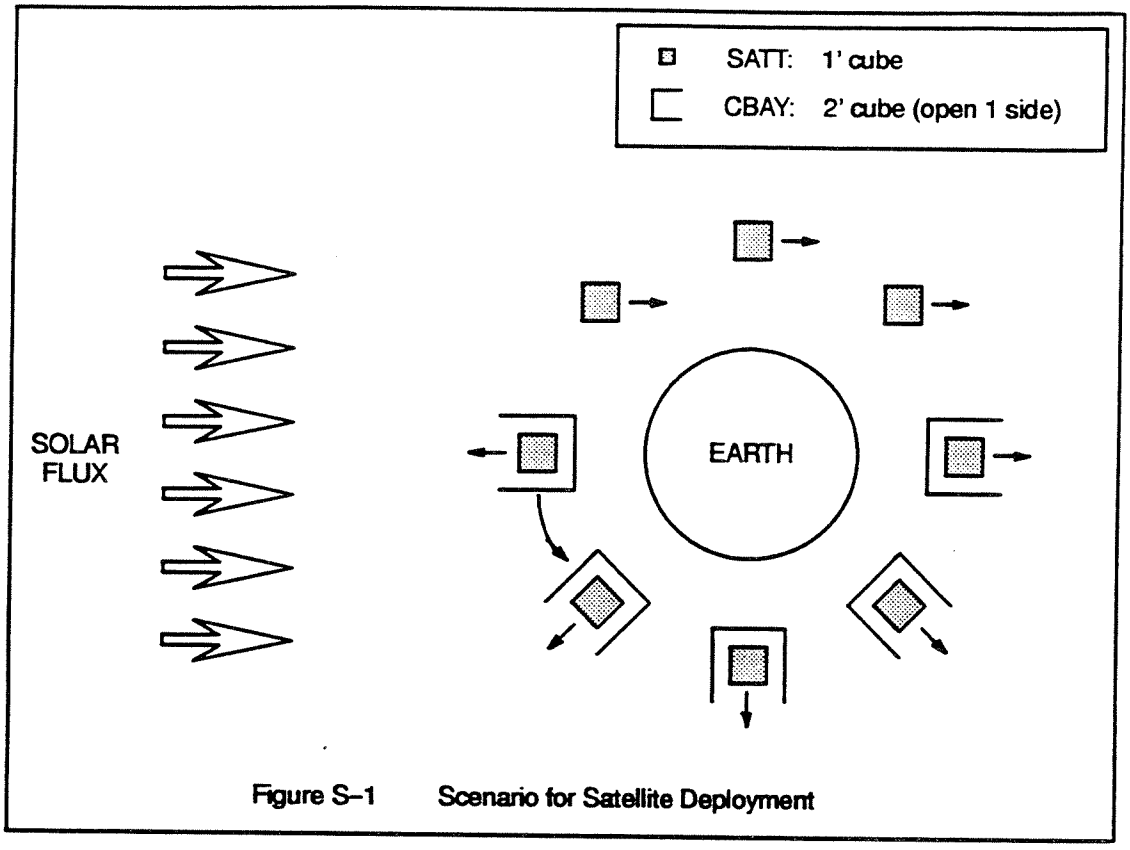


Figure S–1 Scenario for Satellite Deployment

is negligible; a single node is used to characterize both internal and external radiative exchange. Similarly, only one node is chosen to represent the response of the 1 inch solid cube. The thermal capacitance of the cargo bay is ignored; arithmetic nodes are used to represent panels of negligible mass. This would be a good approximation if the cargo bay were insulated internally by multi-layer insulation (MLI).

A separate analysis is required to characterize the radiation exchange between the sun, the earth, the satellite, and the cargo bay, and to determine the heat flux deposition history on the spacecraft surfaces. In this sample, such factors were supplied by the TRASYS II (Thermal Radiation Analysis System, version II) program, which can output SINDA/FLUINT style inputs including submodel distinctions. Most of the input file was therefore computer generated. However, because a discussion of such procedures is beyond the scope of this manual, the reader may disregard the method by which these inputs were derived. The scope of such inputs include:

1. Radiation conductors ("RADKs") between the satellite, the cargo bay, and deep space before deployment,

2. Radiation conductors between the satellite and deep space after deployment (a trivial calculation), and

3. Arrays of impressed heat sources (absorbed earth IR, albedo, and solar) on all exposed surfaces as a functions of time, including initial and final orbital averages.

The radiative exchange within the satellite can be approximated without a TRASYS-type program by using an Effective Radiation Node (ERN) approximation, commonly called an "ERN node." Under this approximation, a nonphysical arithmetic node is introduced to represent the effective temperature of an enclosure, and all internal surfaces are linked to this node (but not directly to each other) by radiation conductors of value $\varepsilon\sigma A$, where ε is the surface IR emissivity, σ is the Stefan-Boltzmann constant, and A is the area of the surface in question. This approximation is very accurate as long as temperature gradients are not severe and no two surfaces have dominant views of each other compared to other surfaces. Because only one node is used to represent the interior solid cube, the radiation conductor for this cube is based on the total surface area (6 in^2).

To summarize the model thus far, there are five arithmetic nodes representing the cargo bay, seven diffusion nodes and one arithmetic node representing the satellite (six surfaces, the interior cube, and the arithmetic ERN node), and one boundary node which is required to represent deep space. The sun and earth appear only as heat sources.

The remaining decisions regarding the model have to do with submodel decisions: Which nodes and conductors belong in how many submodels? To answer these questions, the user must reflect upon the analysis operations to be performed: Which nodes will be active before and after deployment? Which conductors? For this sample, all nodes will be active before deployment, but the cargo bay will disappear after deployment. A different set of radiation conductors will be needed before and after deployment. Therefore, the following decisions are made:

1. The five cargo bay nodes will be contained in a submodel named CBAY along with the entire set of radiation conductors to the interior satellite.
2. The eight satellite nodes will be contained in a submodel named SATT along with the internal radiation conductors.
3. A separate submodel named SPACE will be built that contains no nodes, and only the radiator conductors between SATT and deep space when deployed.
4. For convenience, both SATT and CBAY contain a boundary node representing deep space.

The BUILD and analysis sequences are then:

1. Build SATT and CBAY (neglecting SPACE). Get initial conditions using predeployed orbital average fluxes and a steady-state analysis.
2. Start at the subsolar point and run for half an orbit using a transient analysis.
3. Stop, and rebuild with SPACE and SATT (neglecting CBAY).
4. Complete the orbit using a second transient analysis.
5. Get final orbital averages with a second steady-state run.

For clarity and convenience, deployment is assumed to occur after only a half orbit has elapsed since the time the analysis was started using orbital average initial conditions. In most true orbital analyses, there is no time-independent state that can be used for initial conditions. Orbital averages are used as good initial guesses, and transients must be run for several complete orbits to launder the effects of an unrealistic starting point. The above short cut would only be acceptable if the characteristic thermal time constant (perhaps CSGMAX) were an order of magnitude smaller than the orbital period.

The above breakdown is certainly not the only possible one. Recall that if both ends of an intermodel conductor are not currently built, then that conductor is ignored and no heat is transferred through it. This helps simplify some of the above decisions: once the CBAY submodel is "unbuilt," all conductors to it vanish at the same time independent of the submodel into which they are placed. The user therefore need only make sure that all desired conductors are present in each configuration.

This example illustrates an important and powerful use of submodels and should be studied carefully, especially by users of old SINDA versions. Submodels allow more than just easy merging of models or convenient ways to organize a big model. By the methods similar to those outlined above, whole sets of boundary conditions can be exchanged parametrically in one run. To illustrate this point, a single SINDA/FLUINT run can encompass a rocket sitting on the pad for days, launching, staging, and then deploying a payload. Many powerful tricks are available to the user who has mastered the use of submodels, including features such as DRPMOD and ADDMOD.

S.3 The Input File

The input file is listed immediately following the last section.

OPTIONS DATA—Note that two user files (USER1 and USER2) are named. These file will contain summary listings of the transient event.

CONTROL DATA—English units are chosen, with units of °F (the default). The SIGMA is set to 1.0 (also the default) because this value is already contained within the radiation conductor values in CONDUCTOR DATA. NLOOPS is required for steady-state runs, and is set to 200. OUTPUT is required for transient runs, and is set equal to 1/20th of the orbital period (1.531 hours). DTIMEH, the maximum time step for the transient integration routines, is set to 1/500th of the orbital period. This prevents large time steps in FWDBCK that would skip over important changes in boundary conditions, which in this case are orbital flux variations.

USER DATA—Two global (named) user constants will be used: ISS and ORBIT. ISS is a flag used to tell the logic which steady-state run is being performed: ISS = 1 for the predeployment run, and ISS = 2 is set later to flag the postdeployment run. The logic entered in VARIABLES 1 must know this to determine which set of orbital average fluxes to use. ORBIT contains the orbital period in hours.

NODE DATA—Three thermal submodels are input: SATT, CBAY, and SPACE. SPACE contains no nodes, but *a HEADER NODE DATA record is required to validate the submodel name.* GEN commands are used in SATT and CBAY to generate similar nodes. Separate "3 blanks" commands are used to generate the remaining nodes. Note the usage of arithmetic operations in the node definitions to perform m•Cp calculations and unit conversions. It is recommended that the user leave such calculational trails to make the input file more self-documenting for future analysts, rather than obscuring assumptions by inputting single pre-calculated numbers.

Nodal designations are shown in Figure S–2. The names and types of the elements are as follows:

CBAY.1-5	cargo bay interior (arithmetic)
SATT.11-16	satellite panels (diffusion)
SATT.20	enclosed solid cube (diffusion)
SATT.99	ERN node
SATT.999	deep space (boundary)
CBAY.999	deep space (boundary)

CONDUCTOR DATA—All three thermal submodels have conductors. SPACE is a conductor set whose endpoint nodes are wholly contained within SATT, and is used to swap with those conductors contained in the CBAY submodel. GEN commands would have been more convenient in many places, but these sections were generated by computer. For this reason, the conductor identifiers are not very informative. Note the use of the submodel name to denote nodes not contained within the current submodel (as named on the current HEADER CONDUCTOR DATA record). The Stefan-Boltzmann constant is carried in the calculations for documentation purposes.

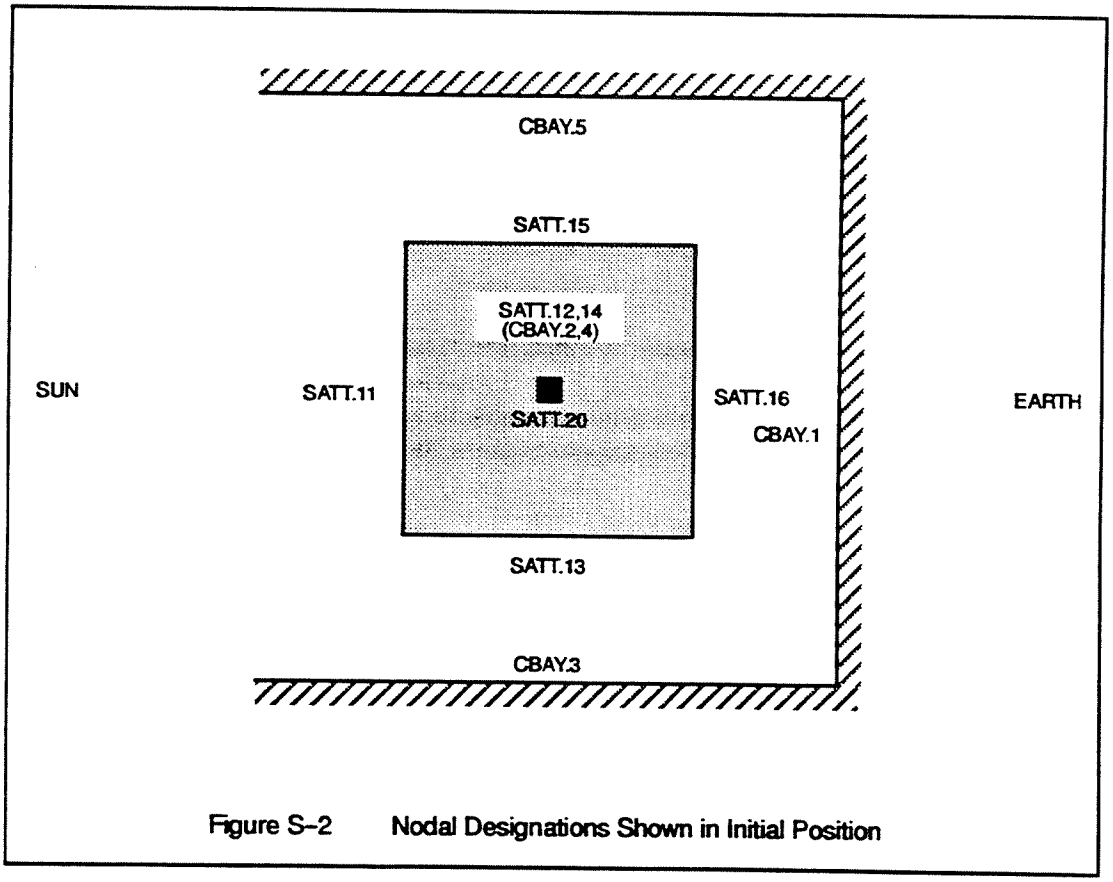


Figure S–2 Nodal Designations Shown in Initial Position

SOURCE DATA—All nodes representing external surfaces have time-varying heat fluxes. These fluxes are input in ARRAY data and the referenced by the TVS option. The program will generate the appropriate interpolation calls in a routine called QVTIME, which will be executed after the VARIABLES 0 call unless the user calls QVTIME explicitly. Note that each TVS call uses the same array to represent time. Note also the use of the surface area as a multiplying factor, leaving the array data in units of flux.

ARRAY DATA—These blocks contain the arrays referenced in the TVS options in SOURCE DATA, and were computer generated as were the TVS commands themselves.

VARIABLES 1—This logic block is called every time step and every steady-state iteration. It is used here to override the TVS-supplied heat rates (Q values) and replace them with orbital average heat rates during the two steady-state runs. Q values in both CBAY and SATT are reset in the variables block for SATT to reduce logic*. Because this routine is only active in steady-states, a check to NSOL is made. Recall that:

FASTIC ..... NSOL=0
STDSTL ..... NSOL=1
FWDBCK ..... NSOL=2
FORWRD ..... NSOL=3

As an aside, note that the fluxes were actually averages for *half* orbits. This violated symmetry for nodes CBAY.3, CBAY.5, SATT.13, and SATT.15. To regain a true orbital average, the fluxes for opposite sides were averaged. This is a minor point made for clarification purposes only.

OUTPUT CALLS—This logic block defines the output to be produced at each output interval. A call is made to the generic temperature printing routine TPRINT, which will write to the file named in OPTIONS DATA after OUTPUT. Also, a compressed line of important temperatures is written to the user file USER1, and important heat rates are similarly written to user file USER2. Note that the first Fortran line avoids reiterating initial conditions in steady-state runs.

CARRAY DATA—A single character array is used to contain an output status line.

OPERATIONS DATA—This block details the analytic operations to be performed. First, the default model name is set to SATT to avoid extra input. Then, the headers in the user summary files are written.

The first operation is to build a configuration named WHOLE consisting of the submodels SATT and CBAY. Any subsequent operations will be performed as if the SPACE submodel were never input. A steady-state is called, and then a transient is run for the first half of the orbit.

In the second part, a new configuration PART is built consisting of SATT and SPACE. The nodes and conductors in CBAY have now disappeared, being replaced by the conductors in SPACE. The orbital transient is completed, and then a final steady-state is run *after setting ISS=2 to signal a new set of average heat rates in VARIABLES 1.*

* Recall that submodel CBAY is not built into the second configuration. No CBAY VARIABLES blocks will be called if CBAY is not active. However, referencing CBAY Q values as done here will not cause problems even when CBAY is not built; those Q values will be changes but never used. This is why the SATT block was used preferentially over the CBAY block.

Remember, even if a model consisted of one and only one submodel (analogous to an old SINDA run), the configuration must be built faithfully with the BUILD command, and the default submodel must still be named explicitly with the DEFMOD command to avoid repeating the submodel name in all translated variables.

S.4 Output Description

The user summary files and selected printings from the OUTPUT file are listed following the input file. Plots of the temperature histories are presented in Figure S–3.

Because only a half-orbit was run before deployment, the transient results are not especially indicative of system response, except to note maximum temperature swings and the relative isolation of the innermost cube.

More intriguing are the initial and final orbital averages, even though they do not necessarily correspond to a realistically achievable state. For instance, the hottest temperature in the entire initial system is node SATT.16 even though that node never has a direct view of the sun. The reason it is the hottest point is that it has an excellent view to reflected sunlight for half of the orbit, but never has a good view to deep space. This counterintuitive result is caused by the optical properties and the view factors, and underscores the importance of codes like TRASYS in a space environment.

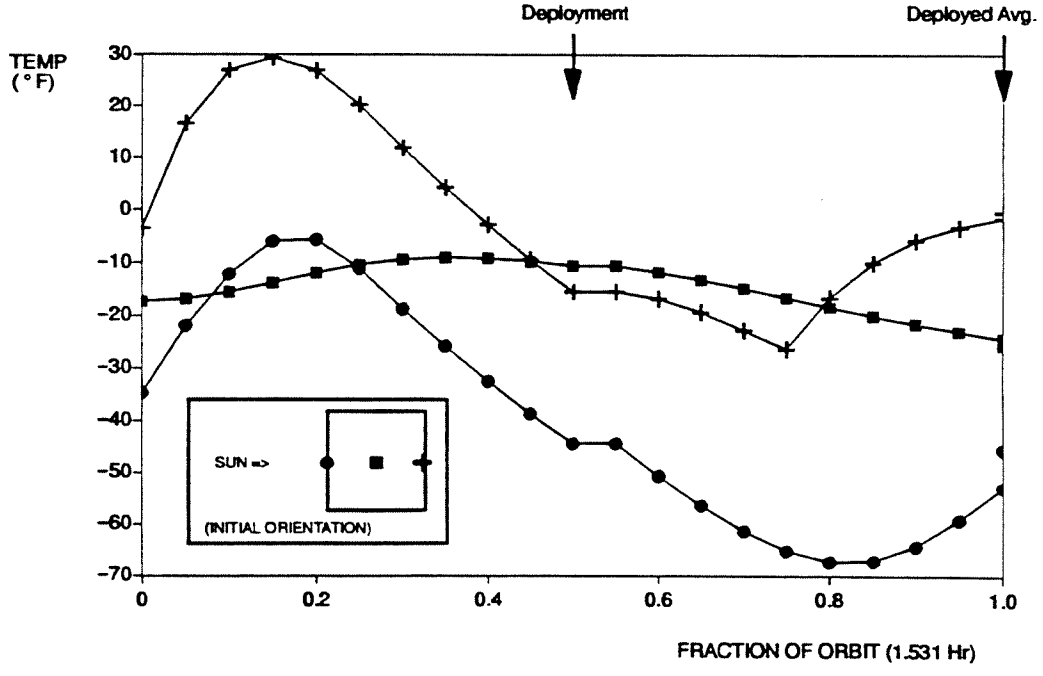


Figure S–3 Initial, Transient, and Final Deployed Temperatures

S.5 A Message to the Novice

A reader new to SINDA may be surprised by the large size of the input deck relative to the simplicity of the system. This is partly due to the "toolbox" nature of SINDA/FLUINT; the system must be described within the context of a generalized network which consists of building blocks and tools that model fundamental phenomena. In this sample, the length of the input file is also due to the dominance of radiation. Unlike conduction, where each node only communicates with a few neighboring nodes, a radiating surface can "see" almost all other surfaces because of reflections.

Also, the novice should not be discouraged by failed first attempts. This program is large and intended to be learned over the course of months, not hours. The user will find the learning process well worth the effort. An experienced user can usually stretch the program to fit almost any system, which is infinitely better than reinventing the wheel with a hardwired, design-specific, user-specific code. Take heart in the fact that even the so-called "expert" who wrote this sample (as well as parts of the code) took about five attempts to get it through the preprocessor, and another five runs to fine-tune the model and the logic to achieve desired results. The moral: start with small models and *think* about the system being modeled.

Input File

```
HEADER OPTIONS DATA
TITLE SINDA SAMPLE PROBLEM DEMONSTRATING SUBMODELS
          OUTPUT    = sinsamp.out
          USER1     = sinsamp.usr
          USER2     = sinsamp.qsr
C
HEADER CONTROL DATA,GLOBAL
          ABSZRO    = -459.67
          SIGMA     = 1.0
          NLOOPS    = 200
          OUTPUT    = 1.531/20.0
          DTIMEH    = 1.531/500.0
C
HEADER USER DATA,GLOBAL
          ISS       = 1
          ORBIT     = 1.531
C
HEADER OPERATIONS DATA
C
DEFMOD SATT
          WRITE(NUSER1,1)'TEMPERATURES'
          WRITE(NUSER2,1)'HEAT RATES'
1         FORMAT('1 SUMMARY TABLE OF SINDA SAMPLE PROBLEM: SIMPLIFIED',
     .    ' SATELLITE DEPLOYMENT  ++ ',A,' ++'//
     .    '  TIME',T12,'CUBE',8X,'OUTSIDE',5X,
     .    'INSIDE',6X,'OTHER SIDE PANELS',29('-'),5X,'STATUS SUMMARY'//)
C
C FIRST HALF OF ORBIT: START WITH ORBITAL AVERAGES
BUILD WHOLE,SATT,CBAY
          UCA1      = 'ORBITAL AVERAGE IN CARGO BAY'
          CALL STDSTL
          UCA1      = 'HALF ORBIT IN CARGO BAY'
          TIMEND    = ORBIT/2.0
          CALL FWDBCK
C
C SECOND HALF OF ORBIT: DEPLOY SATT AND FINISH WITH ORBITAL AVERAGES
BUILD PART,SATT,SPACE
          TIMEND    = ORBIT
          UCA1      = 'HALF ORBIT AFTER DEPLOYED'
          CALL FWDBCK
          UCA1      = 'ORBITAL AVERAGE AFTER DEPLOYED'
          LOOPCT    = 0
          ISS       = 2
          CALL STDSTL
C
```

```
HEADER OUTPUT CALLS,SATT
        IF(NSOL .EQ. 1 .AND. LOOPCT .EQ. 0)RETURN
        CALL TPRINT('ALL')
        WRITE(NUSER1,100)TIMEN,T20,T11,T16,T12,T14,T13,T15,UCA1(1:32)
        WRITE(NUSER2,100)TIMEN,Q20,Q11,Q16,Q12,Q14,Q13,Q15,UCA1(1:32)
100     FORMAT(1X,F6.3,T10,1P,7G12.5,5X,A)
C
HEADER CARRAY DATA,SATT
        1=OUTPUT STATUS LINE
C
HEADER NODE DATA,SATT
C       11      "OUTSIDE" INIT. SUNFACING, FACES SPACE AFTER DEPLOY
C       16      "INSIDE" OPPOSITE 11, FACES SUN AFTER DEPLOY
C       12-15   OTHER SIDES, 12 OPPOSITE 14, 13 OPPOSITE 15
C       20      ENCLOSED 1 INCH CUBE
C       99      ENCLOSURE EFFECTIVE RADIATION NODE
C       999     DEEP SPACE
C
        GEN 11,6,1,70.0,0.214*169.0/8.0/12.0   $ PANELS (1 NODE EACH)
        20,70.0,0.214*169.0/12.0/144.0         $ 1 IN. CUBE (1 NODE)
        99,70.0,-1.0                           $ ERN NODE
        -999,-459.67,0.0                       $ SPACE SINK
C
HEADER CONDUCTOR DATA,SATT
        GEN -11,6,1,11,1,99,0,1.713E-9*0.8  $ PANELS TO ERN
        -20,20,99,1.713E-9*0.8*6.0/144.0    $ ERN TO CUBE
C
HEADER NODE DATA,CBAY
        GEN 1,5,1,70.0,-1.0                 $ CARGO BAY PANELS
        -999,-459.67,0.0                    $ SPACE SINK
C
HEADER NODE DATA,SPACE
```

```
C
C START TRASYS-GENERATED (AND MODIFIED) INPUTS
C
HEADER CONDUCTOR DATA, SPACE
 -  1001,      SATT.11,     SATT.999,  1.71300e-09 * 8.00000e-01
 -  1002,      SATT.12,     SATT.999,  1.71300e-09 * 8.00000e-01
 -  1003,      SATT.13,     SATT.999,  1.71300e-09 * 8.00000e-01
 -  1004,      SATT.14,     SATT.999,  1.71300e-09 * 8.00000e-01
 -  1005,      SATT.15,     SATT.999,  1.71300e-09 * 8.00000e-01
 -  1006,      SATT.16,     SATT.999,  1.71300e-09 * 8.00000e-01
C
HEADER VARIABLES 1,SATT
C
C----- ORBITAL-AVERAGE HEATING RATES - SINDA-85 NODE REFERENCE FORMAT
C        OVERRIDE TVS CALCULATIONS WITH ORBITAL AVERAGES
C        NOTE THAT SIDES 3,5 AND 13,15 ARE AVERAGED BECAUSE THESE
C        FLUXES WERE CALCULATED FOR AN ASYMMETRIC HALF ORBIT
C
         IF(NSOL .NE. 1)RETURN
         IF(ISS .EQ. 1)THEN
                  CBAY.Q1     =  1.41318e+01 * 4.00000e+00
                  CBAY.Q2     =  6.59118e+00 * 4.00000e+00
                  CBAY.Q3     =  1.67453e+01 * 4.00000e+00
                  CBAY.Q4     =  6.59118e+00 * 4.00000e+00
                  CBAY.Q5     =  5.53290e+00 * 4.00000e+00
                  ATEST    = (CBAY.Q3 + CBAY.Q5)*0.5
                  CBAY.Q3     =  ATEST
                  CBAY.Q5     =  ATEST
                  SATT.Q11    =  2.73129e+01 * 1.00000e+00
                  SATT.Q12    =  5.67591e+00 * 1.00000e+00
                  SATT.Q13    =  1.18087e+01 * 1.00000e+00
                  SATT.Q14    =  5.67590e+00 * 1.00000e+00
                  SATT.Q15    =  4.63440e+00 * 1.00000e+00
                  SATT.Q16    =  1.04627e+01 * 1.00000e+00
         ELSE
                  SATT.Q11    =  2.79567e+01 * 1.00000e+00
                  SATT.Q12    =  1.95421e+01 * 1.00000e+00
                  SATT.Q13    =  6.56932e+00 * 1.00000e+00
                  SATT.Q14    =  1.95421e+01 * 1.00000e+00
                  SATT.Q15    =  4.35670e+01 * 1.00000e+00
                  SATT.Q16    =  7.44913e+01 * 1.00000e+00
         ENDIF
C
         ATEST    = (SATT.Q13 + SATT.Q15)*0.5
         SATT.Q13     =  ATEST
         SATT.Q15     =  ATEST
```

```
C
C TRASYS-GENERATED RADKS
C
HEADER CONDUCTOR DATA, CBAY
 -     1,      1,      2,  1.71300e-09 * 4.87484e-01
 -     2,      1,      3,  1.71300e-09 * 4.87484e-01
 -     3,      1,      4,  1.71300e-09 * 4.87484e-01
 -     4,      1,      5,  1.71300e-09 * 4.87484e-01
 -     5,      1,   SATT.11    ,  1.71300e-09 * 4.88200e-03
 -     6,      1,   SATT.12    ,  1.71300e-09 * 5.43616e-02
 -     7,      1,   SATT.13    ,  1.71300e-09 * 5.43616e-02
 -     8,      1,   SATT.14    ,  1.71300e-09 * 5.43616e-02
 -     9,      1,   SATT.15    ,  1.71300e-09 * 5.43616e-02
 -    10,      1,   SATT.16    ,  1.71300e-09 * 5.22813e-01
 -    11,      2,      3,  1.71300e-09 * 4.75932e-01
 -    12,      2,      4,  1.71300e-09 * 1.73520e-01
 -    13,      2,      5,  1.71300e-09 * 4.75932e-01
 -    14,      2,   SATT.11    ,  1.71300e-09 * 3.52032e-02
 -    15,      2,   SATT.12    ,  1.71300e-09 * 1.05428e-02
 -    16,      2,   SATT.13    ,  1.71300e-09 * 5.30734e-02
 -    17,      2,   SATT.14    ,  1.71300e-09 * 5.20893e-01
 -    18,      2,   SATT.15    ,  1.71300e-09 * 5.30734e-02
 -    19,      2,   SATT.16    ,  1.71300e-09 * 5.46980e-02
 -    20,      3,      4,  1.71300e-09 * 4.75932e-01
 -    21,      3,      5,  1.71300e-09 * 1.73520e-01
 -    22,      3,   SATT.11    ,  1.71300e-09 * 3.52032e-02
 -    23,      3,   SATT.12    ,  1.71300e-09 * 5.30734e-02
 -    24,      3,   SATT.13    ,  1.71300e-09 * 5.20893e-01
 -    25,      3,   SATT.14    ,  1.71300e-09 * 5.30734e-02
 -    26,      3,   SATT.15    ,  1.71300e-09 * 1.05428e-02
 -    27,      3,   SATT.16    ,  1.71300e-09 * 5.46980e-02
 -    28,      4,      5,  1.71300e-09 * 4.75932e-01
 -    29,      4,   SATT.11    ,  1.71300e-09 * 3.52032e-02
 -    30,      4,   SATT.12    ,  1.71300e-09 * 5.20893e-01
 -    31,      4,   SATT.13    ,  1.71300e-09 * 5.30734e-02
 -    32,      4,   SATT.14    ,  1.71300e-09 * 1.05428e-02
 -    33,      4,   SATT.15    ,  1.71300e-09 * 5.30734e-02
 -    34,      4,   SATT.16    ,  1.71300e-09 * 5.46980e-02
 -    35,      5,   SATT.11    ,  1.71300e-09 * 3.52032e-02
 -    36,      5,   SATT.12    ,  1.71300e-09 * 5.30734e-02
 -    37,      5,   SATT.13    ,  1.71300e-09 * 1.05428e-02
 -    38,      5,   SATT.14    ,  1.71300e-09 * 5.30734e-02
 -    39,      5,   SATT.15    ,  1.71300e-09 * 5.20893e-01
 -    40,      5,   SATT.16    ,  1.71300e-09 * 5.46980e-02
 -    41,   SATT.11    ,   SATT.12    ,  1.71300e-09 * 1.59630e-03
 -    42,   SATT.11    ,   SATT.13    ,  1.71300e-09 * 1.59630e-03
 -    43,   SATT.11    ,   SATT.14    ,  1.71300e-09 * 1.59630e-03
 -    44,   SATT.11    ,   SATT.15    ,  1.71300e-09 * 1.59630e-03
```

```
-     45,   SATT.11    ,    SATT.16    ,  1.71300e-09 * 5.47783e-04
-     46,   SATT.12    ,    SATT.13    ,  1.71300e-09 * 3.58908e-03
-     47,   SATT.12    ,    SATT.14    ,  1.71300e-09 * 8.23474e-04
-     48,   SATT.12    ,    SATT.15    ,  1.71300e-09 * 3.58908e-03
-     49,   SATT.12    ,    SATT.16    ,  1.71300e-09 * 3.77025e-03
-     50,   SATT.13    ,    SATT.14    ,  1.71300e-09 * 3.58908e-03
-     51,   SATT.13    ,    SATT.15    ,  1.71300e-09 * 8.23474e-04
-     52,   SATT.13    ,    SATT.16    ,  1.71300e-09 * 3.77025e-03
-     53,   SATT.14    ,    SATT.15    ,  1.71300e-09 * 3.58908e-03
-     54,   SATT.14    ,    SATT.16    ,  1.71300e-09 * 3.77025e-03
-     55,   SATT.15    ,    SATT.16    ,  1.71300e-09 * 3.77025e-03
-     56,      1,     999,  1.71300e-09 * 3.53610e-01
-     57,      2,     999,  1.71300e-09 * 7.24377e-01
-     58,      3,     999,  1.71300e-09 * 7.24379e-01
-     59,      4,     999,  1.71300e-09 * 7.24377e-01
-     60,      5,     999,  1.71300e-09 * 7.24375e-01
-     61,   SATT.11    ,    SATT.999   ,  1.71300e-09 * 6.47020e-01
-     62,   SATT.12    ,    SATT.999   ,  1.71300e-09 * 7.34547e-02
-     63,   SATT.13    ,    SATT.999   ,  1.71300e-09 * 7.34546e-02
-     64,   SATT.14    ,    SATT.999   ,  1.71300e-09 * 7.34544e-02
-     65,   SATT.15    ,    SATT.999   ,  1.71300e-09 * 7.34541e-02
-     66,   SATT.16    ,    SATT.999   ,  1.71300e-09 * 2.13119e-02
```

```
C
C----- TIME-VARIABLE HEATING RATES
C
HEADER SOURCE DATA, CBAY
   TVS     1, A     1, A     2,  4.00000e+00
   TVS     2, A     1, A     3,  4.00000e+00
   TVS     3, A     1, A     4,  4.00000e+00
   TVS     4, A     1, A     5,  4.00000e+00
   TVS     5, A     1, A     6,  4.00000e+00
C
HEADER SOURCE DATA, SATT
   TVS    11, A     1, A     7,  1.00000e+00
   TVS    12, A     1, A     8,  1.00000e+00
   TVS    13, A     1, A     9,  1.00000e+00
   TVS    14, A     1, A    10,  1.00000e+00
   TVS    15, A     1, A    11,  1.00000e+00
   TVS    16, A     1, A    12,  1.00000e+00
C
C----- TIME-VARIABLE HEAT RATE TABLES
C
HEADER ARRAY DATA, CBAY
         1$ TIME ARRAY
      0.000e+00, 1.913e-01, 3.827e-01, 4.634e-01, 4.642e-01, 5.740e-01
      7.6525e-01,END$
         2$ HEAT RATE ARRAY
      8.140e+01, 1.583e+01, 0.000e+00, 0.000e+00, 0.000e+00, 0.000e+00
      0.000e+00,END$
         3$ HEAT RATE ARRAY
      2.301e+01, 1.486e+01, 0.000e+00, 0.000e+00, 0.000e+00, 0.000e+00
      0.000e+00,END$
         4$ HEAT RATE ARRAY
      2.301e+01, 5.548e+01, 0.000e+00, 0.000e+00, 0.000e+00, 0.000e+00
      0.000e+00,END$
         5$ HEAT RATE ARRAY
      2.301e+01, 1.486e+01, 0.000e+00, 0.000e+00, 0.000e+00, 0.000e+00
      0.000e+00,END$
         6$ HEAT RATE ARRAY
      2.301e+01, 1.063e+01, 0.000e+00, 0.000e+00, 0.000e+00, 0.000e+00
      0.000e+00,END$
C
```

```
HEADER ARRAY DATA, SATT
         1$ TIME ARRAY
       0.000e+00, 1.913e-01, 3.827e-01, 4.634e-01, 4.642e-01, 5.740e-01
       7.6525e-01,
       7.6526e-01, 9.567e-01, 1.066e+00, 1.067e+00, 1.148e+00, 1.339e+00
       1.531e+00,END$
          7$ HEAT RATE ARRAY
       8.949e+01, 6.451e+01, 0.000e+00, 0.000e+00, 0.000e+00, 0.000e+00
       0.000e+00,
       0.000e+00, 2.209e+00, 9.395e+00, 9.395e+00, 1.795e+01, 5.373e+01
       7.781e+01,END$
          8$ HEAT RATE ARRAY
       1.983e+01, 1.279e+01, 0.000e+00, 0.000e+00, 0.000e+00, 0.000e+00
       0.000e+00,
       1.729e+01, 1.729e+01, 1.729e+01, 1.729e+01, 1.748e+01, 2.249e+01
       2.465e+01,END$
          9$ HEAT RATE ARRAY
       1.983e+01, 3.732e+01, 0.000e+00, 0.000e+00, 0.000e+00, 0.000e+00
       0.000e+00
       1.795e+01, 2.209e+00, 0.000e+00, 0.000e+00, 0.000e+00, 2.776e+00
       2.559e+01,END$
         10$ HEAT RATE ARRAY
       1.983e+01, 1.279e+01, 0.000e+00, 0.000e+00, 0.000e+00, 0.000e+00
       0.000e+00
       1.729e+01, 1.729e+01, 1.729e+01, 1.729e+01, 1.748e+01, 2.249e+01
       2.465e+01,END$
         11$ HEAT RATE ARRAY
       1.983e+01, 8.624e+00, 0.000e+00, 0.000e+00, 0.000e+00, 0.000e+00
       0.000e+00
       1.795e+01, 4.165e+01, 5.156e+01, 5.156e+01, 5.502e+01, 5.472e+01
       2.559e+01,END$
         12$ HEAT RATE ARRAY
       5.577e+01, 1.396e+01, 0.000e+00, 0.000e+00, 0.000e+00, 0.000e+00
       0.000e+00
       5.452e+01, 4.165e+01, 2.751e+01, 1.133e+02, 1.043e+02, 8.876e+01
       8.580e+01,END$
END OF DATA
```

Processor Output

SUMMARY TABLE OF SINDA SAMPLE PROBLEM: SIMPLIFIED SATELLITE DEPLOYMENT ↔ TEMPERATURES ↔

TIME	CUBE	OUTSIDE	INSIDE	OTHER SIDE PANELS				STATUS SUMMARY
0.000	-17.198	-34.644	-3.3499	-19.232	-19.232	-14.254	-14.254	ORBITAL AVERAGE IN CARGO BAY
0.000	-17.198	-34.644	-3.3499	-19.232	-19.232	-14.254	-14.254	HALF ORBIT IN CARGO BAY
0.077	-16.711	-21.754	16.761	-8.2213	-8.2209	-2.3667	-4.4991	HALF ORBIT IN CARGO BAY
0.153	-15.424	-12.024	27.136	3.60718E-02	3.63464E-02	9.9689	1.9242	HALF ORBIT IN CARGO BAY
0.230	-13.634	-5.7930	29.518	5.2786	5.2786	20.931	5.0861	HALF ORBIT IN CARGO BAY
0.306	-11.726	-5.4317	27.100	6.2089	6.2088	23.765	4.8890	HALF ORBIT IN CARGO BAY
0.383	-10.139	-10.828	20.467	2.6992	2.6992	18.408	1.2158	HALF ORBIT IN CARGO BAY
0.459	-9.1755	-18.592	12.106	-3.0041	-3.0041	9.6722	-4.2485	HALF ORBIT IN CARGO BAY
0.536	-8.8054	-25.732	4.4547	-8.5898	-8.5898	1.7364	-9.6391	HALF ORBIT IN CARGO BAY
0.612	-8.9341	-32.342	-2.6039	-14.024	-14.024	-5.5392	-14.914	HALF ORBIT IN CARGO BAY
0.689	-9.4805	-38.497	-9.1603	-19.290	-19.290	-12.262	-20.047	HALF ORBIT IN CARGO BAY
0.766	-10.376	-44.255	-15.286	-24.378	-24.378	-18.514	-25.026	HALF ORBIT IN CARGO BAY
0.766	-10.376	-44.255	-15.286	-24.378	-24.378	-18.514	-25.026	HALF ORBIT AFTER DEPLOYED
0.842	-11.566	-50.545	-16.638	-30.586	-30.586	-26.187	-30.091	HALF ORBIT AFTER DEPLOYED
0.919	-12.999	-56.198	-19.117	-36.208	-36.208	-34.172	-32.984	HALF ORBIT AFTER DEPLOYED
0.995	-14.622	-61.188	-22.606	-41.328	-41.327	-42.329	-34.084	HALF ORBIT AFTER DEPLOYED
1.072	-16.392	-64.997	-26.182	-46.010	-46.010	-49.994	-33.994	HALF ORBIT AFTER DEPLOYED
1.148	-18.165	-67.205	-16.453	-50.150	-50.150	-56.851	-33.191	HALF ORBIT AFTER DEPLOYED
1.225	-19.846	-67.026	-9.8250	-53.531	-53.530	-62.710	-32.272	HALF ORBIT AFTER DEPLOYED
1.301	-21.423	-64.160	-5.5706	-56.079	-56.078	-67.610	-31.537	HALF ORBIT AFTER DEPLOYED
1.378	-22.888	-59.057	-3.1220	-57.921	-57.920	-71.470	-31.227	HALF ORBIT AFTER DEPLOYED
1.454	-24.236	-52.693	-1.4382	-59.304	-59.303	-73.275	-32.865	HALF ORBIT AFTER DEPLOYED
1.531	-25.467	-45.374	-0.26263	-60.305	-60.305	-73.095	-36.378	HALF ORBIT AFTER DEPLOYED
1.531	-68.930	-75.176	-15.752	-89.454	-89.454	-79.897	-79.897	ORBITAL AVERAGE AFTER DEPLOYED

SUMMARY TABLE OF SINDA SAMPLE PROBLEM: SIMPLIFIED SATELLITE DEPLOYMENT ↔ HEAT RATES ↔

TIME	CUBE	OUTSIDE	INSIDE	OTHER SIDE PANELS				STATUS SUMMARY
0.000	0.	27.313	10.463	5.6759	5.6759	8.2215	8.2215	ORBITAL AVERAGE IN CARGO BAY
0.000	0.	89.490	55.770	19.830	19.830	19.830	19.830	HALF ORBIT IN CARGO BAY
0.077	0.	79.669	39.332	17.062	17.062	26.706	15.424	HALF ORBIT IN CARGO BAY
0.153	0.	69.698	22.644	14.252	14.252	33.688	10.951	HALF ORBIT IN CARGO BAY
0.230	0.	52.100	11.275	10.330	10.330	30.141	6.9650	HALF ORBIT IN CARGO BAY
0.306	0.	26.300	5.6913	5.2143	5.2143	15.215	3.5159	HALF ORBIT IN CARGO BAY
0.383	0.	0.49908	0.10800	9.89493E-02	9.89493E-02	0.28872	6.67192E-02	HALF ORBIT IN CARGO BAY
0.459	0.	0.	0.	0.	0.	0.	0.	HALF ORBIT IN CARGO BAY
0.536	0.	0.	0.	0.	0.	0.	0.	HALF ORBIT IN CARGO BAY
0.612	0.	0.	0.	0.	0.	0.	0.	HALF ORBIT IN CARGO BAY
0.689	0.	0.	0.	0.	0.	0.	0.	HALF ORBIT IN CARGO BAY
0.766	0.	0.	0.	0.	0.	0.	0.	HALF ORBIT IN CARGO BAY
0.766	0.	2.76971E-03	54.504	17.290	17.290	17.930	17.980	HALF ORBIT AFTER DEPLOYED
0.842	0.	0.87282	49.435	17.290	17.290	11.730	27.314	HALF ORBIT AFTER DEPLOYED
0.919	0.	1.7517	44.314	17.290	17.290	5.4676	36.744	HALF ORBIT AFTER DEPLOYED
0.995	0.	4.6363	36.874	17.290	17.290	1.4628	44.997	HALF ORBIT AFTER DEPLOYED
1.072	0.	9.7298	112.95	17.297	17.297	0.	51.695	HALF ORBIT AFTER DEPLOYED
1.148	0.	17.815	104.44	17.477	17.477	0.	54.965	HALF ORBIT AFTER DEPLOYED
1.225	0.	32.050	98.176	19.454	19.454	1.0940	54.902	HALF ORBIT AFTER DEPLOYED
1.301	0.	46.391	91.948	21.462	21.462	2.2066	54.782	HALF ORBIT AFTER DEPLOYED
1.378	0.	58.417	88.184	22.910	22.910	7.2165	49.050	HALF ORBIT AFTER DEPLOYED
1.454	0.	68.018	87.004	23.772	23.772	16.312	37.436	HALF ORBIT AFTER DEPLOYED
1.531	0.	77.618	85.824	24.633	24.633	25.408	25.822	HALF ORBIT AFTER DEPLOYED
1.531	0.	27.957	74.491	19.542	19.542	25.068	25.068	ORBITAL AVERAGE AFTER DEPLOYED

MODEL = SINDA85
STDSTL

SINDA SAMPLE PROBLEM DEMONSTRATING SUBMODELS

SUBMODEL NAME = SATT

		CALCULATED		ALLOWED	
MAX DIFF DELTA T PER ITER	DRLXCC(SATT	14)= -8.575439E-03	VS.	DRLXCA= 1.000000E-02	
MAX ARITH DELTA T PER ITER	ARLXCC(SATT	99)= -3.692627E-03	VS.	ARLXCA= 1.000000E-02	
MAX SYSTEM ENERGY BALANCE	EBALSC	= -1.385593E-02	VS.	EBALSA = ESUMIS	= 0.655705
				EBALSA= 1.000000E-02	
ENERGY INTO AND OUT OF SYS	ESUMIS	= 65.5705		ESUMOS= 57.1271	
MAX NODAL ENERGY BALANCE	EBALNC(SATT	14)= -2.505779E-03	VS.	EBALNA= 0.	
NUMBER OF ITERATIONS	LOOPCT	= 22	VS.	NLOOPS= 200	
PROBLEM TIME	TIMEN	= 0.	VS.	TIMEND= 0.	

DIFFUSION NODES IN INPUT NODE NUMBER ORDER

T 11= -34.644 T 12= -19.232 T 13= -14.254 T 14= -19.232 T 15= -14.254 T 16= -3.3499
T 20= -17.198

ARITHMETIC NODES IN INPUT NODE NUMBER ORDER

T 99= -17.202

HEATER NODES IN INPUT NODE NUMBER ORDER

←→NONE←→

BOUNDARY NODES IN INPUT NODE NUMBER ORDER

T 999= -459.67

MODEL = SINDA85
STDSTL

SINDA SAMPLE PROBLEM DEMONSTRATING SUBMODELS

SUBMODEL NAME = CBAY

		CALCULATED		ALLOWED	
MAX DIFF DELTA T PER ITER	DRLXCC(	0)= 0.	VS.	DRLXCA= 1.000000E-02	
MAX ARITH DELTA T PER ITER	ARLXCC(CBAY	2)= -2.105713E-03	VS.	ARLXCA= 1.000000E-02	
MAX SYSTEM ENERGY BALANCE	EBALSC	= -3.501892E-03	VS.	EBALSA = ESUMIS	= 1.98369
				EBALSA= 1.000000E-02	
ENERGY INTO AND OUT OF SYS	ESUMIS	= 198.369		ESUMOS= 206.830	
MAX NODAL ENERGY BALANCE	EBALNC(CBAY	1)= -1.747131E-03	VS.	EBALNA= 0.	
NUMBER OF ITERATIONS	LOOPCT	= 22	VS.	NLOOPS= 200	
PROBLEM TIME	TIMEN	= 0.	VS.	TIMEND= 0.	

DIFFUSION NODES IN INPUT NODE NUMBER ORDER

←→NONE←→

ARITHMETIC NODES IN INPUT NODE NUMBER ORDER

T 1= -1.0079 T 2= -27.782 T 3= -16.891 T 4= -27.782 T 5= -16.891

HEATER NODES IN INPUT NODE NUMBER ORDER

←→NONE←→

BOUNDARY NODES IN INPUT NODE NUMBER ORDER

T 999= -459.67

SUBMODEL NAME = SATT

ARLXCA =	1.000000E-02	ARLXCC =	0.	ATMPCA =	1.000000E+30	ATMPCC =	0.	BACKUP =	0.		
CSGFAC =	0.	CSGMAX =	0.	CSGMIN =	0.	DRLXCA =	1.000000E-02	DRLXCC =	0.		
DTIMEB =	3.062000E-03	DTIMEI =	0.	DTIMEL =	0.	DTIMEU =	0.	DTMPCA =	1.000000E+30		
DTMPCC =	0.	EBALNA =	0.	EBALNC =	-2.505779E-03	EBALSA =	1.000000E-02	EBALSC =	0.		
ESUMIS =	65.5705	ESUMOS =	57.1271	EXTLIM =	50.0000	ITHOLD =	10	NARLXN =	0		
NATMPN =	0	NCGMAN =	0	NCSGMN =	0	NDRLXN =	0	NDTMPN =	0		
NEBALN =	0	NLOOPT =	100	ITEROT =	0	ITERXT =	3	OPEITR =	0		
OUTPUT =	7.655000E-02										

MODEL = SINDA85
FWDBCK

SINDA SAMPLE PROBLEM DEMONSTRATING SUBMODELS

SUBMODEL NAME = SATT

	CALCULATED		ALLOWED
MAX DIFF DELTA T PER ITER	DRLXCC(SATT	16)=-6.103516E-05	VS. DRLXCA= 1.000000E-02
MAX ARITH DELTA T PER ITER	ARLXCC(SATT	99)= 0.	VS. ARLXCA= 1.000000E-02
MAX DIFF DEL T PER TIME STEP	DTMPCC(SATT	13)=-0.241821	VS. DTMPCA= 1.000000E+30
MAX ARITH DEL T PER TIME STEP	ATMPCC(SATT	99)=-0.215149	VS. ATMPCA= 1.000000E+30
MIN STABILITY CRITERIA	CSGMIN(SATT	16)= 0.451018	
MAX STABILITY CRITERIA	CSGMAX(SATT	20)= 1.06072	
NUMBER OF ITERATIONS	LOOPCT	= 4	VS. NLOOPT= 100
PROBLEM TIME	TIMEN	= 0.765500	VS. TIMEND= 0.765500
MEAN PROBLEM TIME	TIMEM	= 0.763970	
AVERAGE TIME STEP USED SINCE LAST OUTPUT		= 3.062000E-03	VS. DTIMEI= 0.

DIFFUSION NODES IN INPUT NODE NUMBER ORDER

T 11= -44.255 T 12= -24.378 T 13= -18.514 T 14= -24.378 T 15= -25.026 T 16= -15.286
T 20= -10.376

ARITHMETIC NODES IN INPUT NODE NUMBER ORDER

T 99= -24.913

HEATER NODES IN INPUT NODE NUMBER ORDER

↔NONE↔

BOUNDARY NODES IN INPUT NODE NUMBER ORDER

T 999= -459.67

MODEL = SINDA85
FWDBCK

SINDA SAMPLE PROBLEM DEMONSTRATING SUBMODELS

SUBMODEL NAME = CBAY

	CALCULATED		ALLOWED
MAX DIFF DELTA T PER ITER	DRLXCC(	0)= 0.	VS. DRLXCA= 1.000000E-02
MAX ARITH DELTA T PER ITER	ARLXCC(CBAY	1)= 3.570557E-03	VS. ARLXCA= 1.000000E-02
MAX DIFF DEL T PER TIME STEP	DTMPCC(	0)= 0.	VS. DTMPCA= 1.000000E+30
MAX ARITH DEL T PER TIME STEP	ATMPCC(CBAY	1)=-0.192810	VS. ATMPCA= 1.000000E+30
MIN STABILITY CRITERIA	CSGMIN(	0)= 1.000000E+30	
MAX STABILITY CRITERIA	CSGMAX(	0)=-1.000000E+30	
NUMBER OF ITERATIONS	LOOPCT	= 4	VS. NLOOPT= 100
PROBLEM TIME	TIMEN	= 0.765500	VS. TIMEND= 0.765500
MEAN PROBLEM TIME	TIMEM	= 0.763970	
AVERAGE TIME STEP USED SINCE LAST OUTPUT		= 3.062000E-03	VS. DTIMEI= 0.

DIFFUSION NODES IN INPUT NODE NUMBER ORDER

↔NONE↔

ARITHMETIC NODES IN INPUT NODE NUMBER ORDER

T 1= -76.881 T 2= -88.711 T 3= -87.397 T 4= -88.711 T 5= -89.078

HEATER NODES IN INPUT NODE NUMBER ORDER

↔NONE↔

BOUNDARY NODES IN INPUT NODE NUMBER ORDER

T 999= -459.67

MODEL = SINDA85 SINDA SAMPLE PROBLEM DEMONSTRATING SUBMODELS
FWDBCK

SUBMODEL NAME = SATT

		CALCULATED			ALLOWED	
MAX DIFF DELTA T PER ITER	DRLXCC(SATT	16)=	3.051758E-05	VS. DRLXCA=	1.000000E-02	
MAX ARITH DELTA T PER ITER	ARLXCC(SATT	99)=	3.051758E-05	VS. ARLXCA=	1.000000E-02	
MAX DIFF DEL T PER TIME STEP	DTMPCC(SATT	11)=	0.308594	VS. DTMPCA=	1.000000E+30	
MAX ARITH DEL T PER TIME STEP	ATMPCC(SATT	99)=	2.447510E-02	VS. ATMPCA=	1.000000E+30	
MIN STABILITY CRITERIA	CSGMIN(SATT	16)=	0.634701			
MAX STABILITY CRITERIA	CSGMAX(SATT	20)=	1.19263			
NUMBER OF ITERATIONS	LOOPCT	=	2	VS. NLOOPT=	100	
PROBLEM TIME	TIMEN	=	1.53100	VS. TIMEND=	1.53100	
MEAN PROBLEM TIME	TIMEM	=	1.52947			
AVERAGE TIME STEP USED SINCE LAST OUTPUT		=	3.062000E-03	VS. DTIMEI=	0.	

DIFFUSION NODES IN INPUT NODE NUMBER ORDER

T 11= -45.374 T 12= -60.305 T 13= -73.095 T 14= -60.305 T 15= -36.378 T 16=-0.26263
T 20= -25.467

ARITHMETIC NODES IN INPUT NODE NUMBER ORDER

T 99= -43.750

HEATER NODES IN INPUT NODE NUMBER ORDER
↔NONE↔

BOUNDARY NODES IN INPUT NODE NUMBER ORDER

T 999= -459.67

PROBLEM A: FILLING TOY BALLOONS

This problem simulates the simultaneous filling of three toy balloons using a pressurized air tank. This problem was chosen to demonstrate the following FLUINT features:

1. the properties of gas-filled tanks
2. the use of compliances (albeit in an extreme case)
3. duplication options
4. user logic for specialized output
5. user logic for customized solutions
6. user-defined gases
7. choked valves
8. the broad range of application of the FLUINT code

A.1 Problem Description

Three identical balloons are to be filled from one air tank (see Figure A–1). There is a hose leading from the supply tank to a tee. Three hand valves are attached to the tee, one for each balloon. The tank is fully charged to 22 atmospheres gauge, the balloons are resting at atmospheric pressure, and the entire system is at room temperature (70°F). At time zero, the hand valves are opened simultaneously and the balloons begin to fill. The simulation ends when each balloon has reached a volume of 1 ft^3. The entire system is assumed adiabatic. The relevant characteristics are:

Balloon:		
	Number of balloons	3
	Initial balloon diameter (spherical)	2 in.
	Final balloon volume	1 ft^3
	Initial thickness	0.005 in.
	Modulus	1500 psi
Tank:		
	Volume	3 ft^3
	Initial pressure	22 atm.
Valve:		
	Open flow area	0.002 sq. in.
	Head loss K-factor	25.0
Hose:		
	Diameter (inner)	1/8 in.
	Length	5 ft.

A.2 A FLUINT Model

Most modeling choices are clear. Logically, a rigid FLUINT "tank" will be used to model the supply tank. Compliant tanks will be used to model the balloons; this is stretching the intended use of the compliance factor, but is still perfectly valid. Because hydrodynamic inertia is negligible in a gas, an STUBE connector will be used to include the pressure drop in the hose.

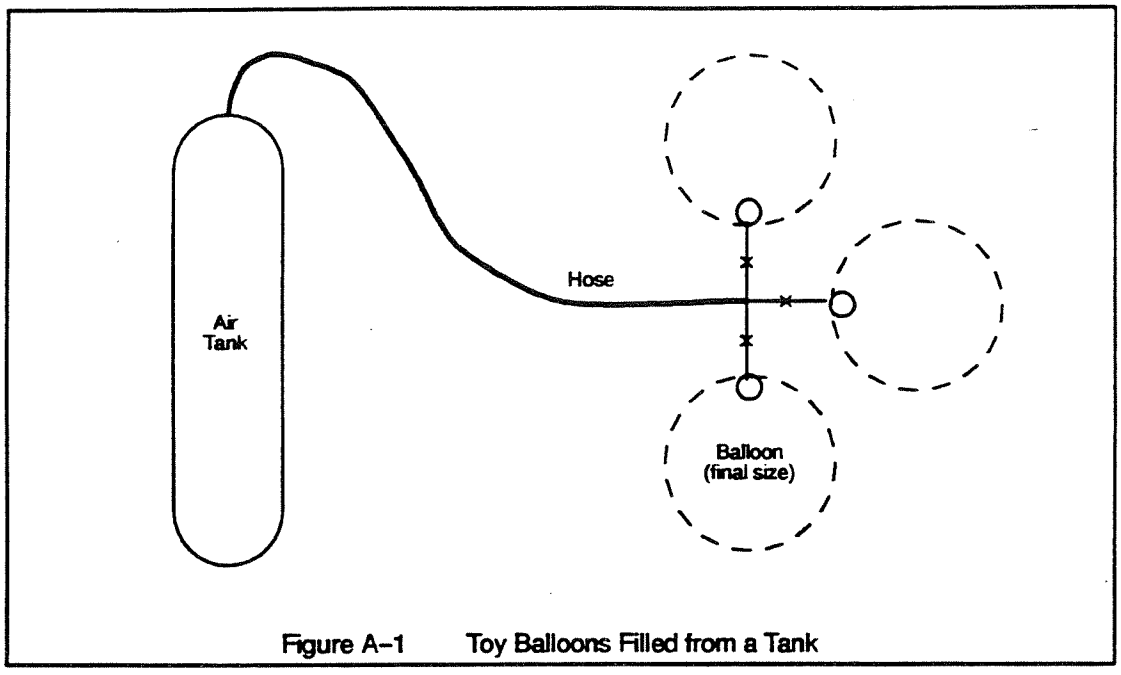


Figure A-1 Toy Balloons Filled from a Tank

The symmetry in the system is obvious. An upstream duplication factor (DUPI) of 3.0 applied to a valve will simulate the effects on the tank of filling three balloons, while only one valve and one balloon need be input and analyzed.

The modeling choice for the valve itself is less clear. At first, it might seem desirable to use a CTLVLV connector that is initially closed (negative FK), and then opened at time zero. The user would then use a positive FK value of 25.0 when the valve is open. However, the large pressure difference between the supply tank (whose pressure should not diminish significantly during the fill) and the balloons (whose pressure will remain near atmospheric) means that the valve is almost certainly choked—the Mach number in the valve throat cannot be greater than unity. For these reasons, a simple LOSS connector is chosen to model the valve and the CHOKER simulation routine will be used in the FLOGIC 1 block to reflect the choked mass flowrate.* (CHOKER requires the use of a STUBE or LOSS family connector because of its use of the parameter AFTH, or throat area.)

As will be shown, the valve remains choked throughout this simulation. If this were always true, an MFRSET connector could have been used to model the valve, with the SMFR set according to a call to CHKCHL.

Choosing a correct value for the balloon compliance requires some thought. For the purposes of this example, the compliance of the balloon will be estimated roughly. Note that the thickness and the modulus noted above are rough estimates for a rubber balloon. Assuming the modulus remains constant (which it doesn't), the compliance should scale similarly to a flexible line: D/(Et). Note that as the balloon stretches, the thickness decreases in approximate proportion to the inverse of the diameter squared. This means that the compliance will increase in proportion to the volume, which fits with experience. This can be taken into account in FLOGIC

* Sudden valve changes are not so easily tolerated in a liquid model.

0 using the current volume of the balloon to update the compliance. The initial value of the compliance actually used is 10D/(Et). The factor of ten was applied after an initial run showed the compliance to be too low by an order of magnitude; the final balloon pressure was too high by at least an order of magnitude.

A.3 The Input File

The input file is listed immediately following the last section.

OPTIONS DATA—Note that a user file (USER1) is named. This file will contain a summary listing of the transient event.

CONTROL DATA—English units are chosen, with units of °F and psig. The value of OUTPTF is required, but the value supplied here will be overwritten later. Because the time it takes to fill the balloon is unknown beforehand, logic will be added to call for printouts and to end the simulation at the appropriate times.

USER DATA—The initial COMP/VOL product is input, and variables to be used in logic blocks are reserved (this is necessary only if the DEBUG option is used). Note that multiplying CVINIT times the balloon volume yields the compliance.

FLOW DATA—Only one submodel is used: a fluid submodel named FILLIT. The working fluid is "8000," meaning a user-defined gas described elsewhere in the input. The default state is given as the tank temperature and pressure, the balloon initial conditions will override this pressure. The default initial flowrate is zero.

The names and types of the elements are as follows:

TANK 1	supply tank
STUBE 10	hose
JUNC 2	tee between hose and valves
LOSS 20	a valve
TANK 3	a single balloon

The DUPI for the LOSS connector is 3.0. The initial balloon compliance will be immediately overridden by user logic in FLOGIC 0, as described next.

FLOGIC 0—This logic block contains the calculation of the balloon compliance as a function of volume.

FLOGIC 1—This logic block contains the call to CHOKER to simulate the choking of the valve. Zero hysteresis (the last argument in CHOKER) is applied.

The flowrate through the LOSS connector will be set to the choked limit when the pressure ratio exceeds the critical ratio. If the pressure ratio were ever to fall below the critical value as the supply tank empties, the LOSS would resume normal operation (i.e., obey a K–factor head loss relationship).

FLOGIC 2—This logic block contains two calculations: the adjustment of the output interval to print at 5% increments of the balloon volume, and the signaling of the end of the run.

FPROP DATA—This data block defines air as perfect gas number 8000. All properties are assumed constant, and are input in SI units (conversions are automatic).

OUTPUT CALLS—This logic block defines the output to be produced at each output interval. Note that a compressed line of important data is written to the user file USER1.

OPERATIONS DATA—Even though there is only one submodel, the configuration must be built faithfully. Also, the default model name is set to avoid extra input. As a check on the conservation of total mass, the initial and final masses are calculated and written out. A header for the user summary file is also written here. The only analysis operation is a transient using FORWRD. No steady-state solution is needed. Also, note that FORWRD and FWDBCK are identical for fluid submodels. Finally, the endpoint conditions including final mass are written to the user summary file.

A.4 Output Description

The user summary file and selected printings from the OUTPUT file are listed following the input file.

With regard to the output formats and the program response, note that the time step was limited by the volume change of the balloon. Over a volume increase of more than 400 times, the number of time steps required was between 100 and 200. Note that the total system mass was conserved to better than 5 decimal places throughout the integration of the equations over hundreds of time intervals.

With regard to the physical response of the system, note that almost all of the pressure rise in the balloon occurs within the first few microseconds of the event. The temperature across the valve does not change significantly (there is no Joule-Thompson effect for a perfect gas); the temperature drop in the balloon is due to the temperature drop in the supply tank as its pressure drops. In fact, inspection of the full OUTPUT file reveals that the balloon temperature initially rose by a few tenths of a degree before starting to descend. This was caused by the compression of the gas against the higher compliance of the small balloon. The total fill time was about 6.5 seconds; this seems a little slow—the valve area may be too small.

Although this response seems intuitively correct, no experimental data was available. A more comprehensive discussion of this subject can be found in Jearl Walker's Amateur Scientist column in the December 1989 issue of *Scientific American*. This column contains more complete references to other work on toy balloons, and points out the importance of a constant pressure source versus a constant mass flowrate source. Because of choking, this particular sample problem represents the latter rather than the former even though a large supply bottle is used. Walker suggests that the pressure should actually peak in such constant flowrate cases, and then decrease as the radius of curvature increases. Decreasing pressure with increasing volume implies negative compliances in the FLUINT model. Negative compliances are legal although not often used, and can cause instabilities if not used carefully. Unfortunately, this column did not present enough specific information to update the membrane flexibility model used in this sample problem. Fortunately, an accurate model is unnecessary for demonstration purposes.

Input File

```
HEADER OPTIONS DATA
TITLE FLUINT SAMPLE PROBLEM 1 - FILLING AIR BALLOONS
         MODEL    = BALLOON
         USER1    = balloon.usr
         OUTPUT   = balloon.out
C
HEADER CONTROL DATA,GLOBAL
         UID      = ENG
         ABSZRO   = -460.0
         PATMOS   = -14.7
         OUTPTF   = 0.1
C
HEADER USER DATA, GLOBAL
         TOTMAS           = 0.0                $ TOTAL MASS IN SYS.
C
C DIA = 2", THK = 5 MILS, MODULUS = 1500 PSI, INIT VOL = 0.002424
C 10.0 IS A FUDGE FACTOR FOUND BY TRIAL RUNS TO
C BRING PRESSURES INTO AN INTUITIVE RANGE FOR BALLOONS
C
         CVINIT = 10.0*2.0/1500.0/0.005/0.002424  $ COMP/VOL INIT
C
HEADER FLOW DATA,FILLIT,FID=8000                  $ SUBMODEL DEFINITION
C
C DEFAULT CONDITIONS (THOSE OF SUPPLY TANK)
C
LU DEF, PL = 22.0*14.7, TL = 70.0, XL = 1.0
PA DEF, FR = 0.0, DH = 0.125/12.0                 $ 1/8" DIA
C
LU TANK,1,          VOL      = 3.0                $ SUPPLY TANK
PA CONN,10,1,2,     DEV      = STUBE              $ HOSE
                    TLEN     = 5.0                $ 5' LONG
LU JUNC,2                                         $ TEE TO BALLOONS
PA CONN,20,2,3,     DUPI     = 3.0                $ VALVE TO 3 BALLOONS
                    DEV      = LOSS
                    FK       = 25.0               $ K-FACTOR, UNCHOKED
                    AFTH     = 0.002/144.0        $ THROAT AREA
LU TANK,3,          VOL      = 0.002424           $ A BALLOON, 2" DIA
                    PL       = 0.0                $ AT ATMOSPHERIC PRESS.
                    COMP     = 1.0                $ FAKE INIT VALUE
C
```

```
HEADER FLOGIC 0, FILLIT                              $ LOGIC BLOCK
C
C THIS LOGIC CHANGES BALLOON COMPLIANCE AS IT FILLS
C PROPORTIONAL TO D/t, t PROPORTIONAL TO 1/D**2, SO
C COMPLIANCE PROPORTIONAL TO VOLUME (DV/DP = CONSTANT)
C
        COMP(3)          = CVINIT*VOL(3)
C
HEADER FLOGIC 1, FILLIT                              $ LOGIC BLOCK
C
C SIMULATES CHOKING IF VALVE IS CHOKED.
C
        CALL CHOKER('FILLIT',20,1,0.0)
C
HEADER FLOGIC 2, FILLIT                              $ LOGIC BLOCK
C
C THIS LOGIC ADJUSTS THE OUTPUT INTERVAL FOR ABOUT EVERY 0.05 CUFT
C
        IF(FR20 .LE. 0.0)THEN
                OUTPTF   = 0.05
        ELSE
                OUTPTF   = 0.05*DL3/FR20             $ EST 0.05 INTERVALS
        ENDIF
        IF(VOL(3) .GE. 1.0) TIMEND = TIMEN           $ STOP AT 1 CUFT
C
HEADER FPROP DATA, 8000, SI, 0.0                     $ DESCRIBE AIR
        RGAS     = 8314.34/28.97
        CP       = 1002.0
        K        = 7.4E-3
        V        = 5.38E-6
C
HEADER OUTPUT CALLS, FILLIT                          $ OUTPUT OPERATIONS
        CALL LMPTAB('FILLIT')                        $ TEMPS & PRESS
        CALL LMXTAB('FILLIT')                        $ VOLS & COMPS
        CALL PTHTAB('FILLIT')                        $ FLOWRATES
C
C NOW, THE CONDENSED TABLE OF PARAMETERS ON USER FILE
C
        WRITE(NUSER1,10) 3600.0*TIMEN,PL(1),FR(20)
     .           ,PL(3),VOL(3),COMP(3)
F10     FORMAT(1X,6(1PG13.5,7X))
C
```

```
HEADER OPERATIONS DATA                                    $ LIST OPERATIONS
BUILDF CONFIG,FILLIT
C
DEFMOD FILLIT
         TIMEND              = 10.0
         TOTMAS    = VOL1*DL1 + DUPI20*VOL3*DL3
         WRITE(NUSER1,10) TOTMAS
C                              .
         CALL FORWRD
C
         TOTMAS              = VOL1*DL1 + DUPI20*VOL3*DL3
         WRITE(NUSER1,20) TOTMAS,TL1,TL3
100      CONTINUE
FSTART
10       FORMAT(' THE INITIAL SYSTEM MASS IS:',T40,1PG13.5///
     .   '   TIME (SEC)',T25,'P SUPPLY',T45,'FLOWRATE',
     .   T65,'P BALLOON',T85,'V BALLOON',T105,'COMPLIANCE'/)
20       FORMAT(//' THE FINAL SYSTEM MASS IS:',T40,1PG13.5/
     .   ' THE FINAL SUPPLY TEMPERATURE IS:',T40,G13.5/
     .   ' THE FINAL BALLOON TEMPERATURE IS:',T40,G13.5)
FSTOP
C
END OF DATA
```

Processor Output

THE INITIAL SYSTEM MASS IS: 5.1737

TIME (SEC)	P SUPPLY	FLOWRATE	P BALLOON	V BALLOON	COMPLIANCE
0.	323.40	0.	0.	2.42400E-03	2.6667
0.31106	322.35	44.220	0.35348	5.26109E-02	57.878
0.62406	321.30	44.098	0.36213	0.10250	112.76
0.93840	320.24	43.976	0.36512	0.15241	167.67
1.2539	319.19	43.853	0.36664	0.20232	222.57
1.5706	318.14	43.731	0.36755	0.25222	277.47
1.8884	317.08	43.608	0.36816	0.30211	332.35
2.2073	316.03	43.486	0.36860	0.35198	387.21
2.5273	314.98	43.363	0.36893	0.40181	442.04
2.8484	313.93	43.241	0.36919	0.45164	496.86
3.1706	312.87	43.118	0.36940	0.50145	551.65
3.4939	311.82	42.996	0.36956	0.55124	606.42
3.8182	310.77	42.873	0.36970	0.60100	661.16
4.1437	309.72	42.750	0.36982	0.65073	715.88
4.4703	308.67	42.628	0.36992	0.70046	770.58
4.7980	307.62	42.505	0.37001	0.75015	825.24
5.1268	306.57	42.383	0.37009	0.79982	879.89
5.4567	305.52	42.260	0.37016	0.84947	934.51
5.7877	304.48	42.149	0.37022	0.89906	988.98
6.1198	303.43	42.015	0.37027	0.94865	1043.6
6.4531	302.38	41.903	0.37032	0.99820	1098.0
6.6197	301.86	41.840	0.37034	1.0229	1098.1

THE FINAL SYSTEM MASS IS: 5.1737
THE FINAL SUPPLY TEMPERATURE IS: 60.115
THE FINAL BALLOON TEMPERATURE IS: 64.983

MODEL = BALLOON FLUINT SAMPLE PROBLEM 1 - FILLING AIR BALLOONS
FORWRD

FLUID SUBMODEL NAME = FILLIT ; FLUID NO. = 8000

MAX TIME STEP = 0. ; LIMITING MESSAGE = (NO SOLUTION STEP TAKEN YET)
LAST TIME STEP = 0. VS. DTMAXF/DTMINF = 1.000000E+30 / 0. ; AVERAGE TIME STEP = 0.
PROBLEM TIME TIMEN = 0. VS. TIMEND = 10.0000

LUMP PARAMETER TABULATION FOR SUBMODEL FILLIT

LUMP	TYPE	TEMP	PRESSURE	QUALITY	DENSITY	ENTHALPY	HEAT RATE	MASS RATE	ENERGY RATE
1	TANK	70.00	323.4	1.000	1.724	126.8	0.	0.	0.
3	TANK	70.00	0.	1.000	7.4973E-02	126.8	0.	0.	0.
2	JUNC	70.00	323.4	1.000	1.724	126.8	0.	0.	0.

MODEL = BALLOON
FORWRD

FLUINT SAMPLE PROBLEM 1 - FILLING AIR BALLOONS

FLUID SUBMODEL NAME = FILLIT ; FLUID NO. = 8000

MAX TIME STEP = 0. ; LIMITING MESSAGE = (NO SOLUTION STEP TAKEN YET)
LAST TIME STEP = 0. VS. DTMAXF/DTMINF = 1.000000E+30 / 0. ; AVERAGE TIME STEP = 0.
PROBLEM TIME TIMEN = 0. VS. TIMEND = 10.0000

LUMP EXTRA PARAMETER TABULATION FOR SUBMODEL FILLIT
BODY FORCE ACCELERATION COMPONENTS (X,Y,Z): 0. , 0. , 0.

LUMP	TYPE	CX	CY	CZ	NO. PATHS	HOLDS	MASS	VOLUME	COMPLIANCE	VOLUME RATE
1	TANK	0.	0.	0.	1	NONE	5.173	3.000	0.	0.
3	TANK	0.	0.	0.	1	NONE	1.8173E-04	2.4240E-03	2.667	0.
2	JUNC	0.	0.	0.	2	NONE				

MODEL = BALLOON
FORWRD

FLUINT SAMPLE PROBLEM 1 - FILLING AIR BALLOONS

FLUID SUBMODEL NAME = FILLIT ; FLUID NO. = 8000

MAX TIME STEP = 0. ; LIMITING MESSAGE = (NO SOLUTION STEP TAKEN YET)
LAST TIME STEP = 0. VS. DTMAXF/DTMINF = 1.000000E+30 / 0. ; AVERAGE TIME STEP = 0.
PROBLEM TIME TIMEN = 0. VS. TIMEND = 10.0000

PATH PARAMETER TABULATION FOR SUBMODEL FILLIT

PATH	TYPE	LMP 1	LMP 2	DUP I	DUP J	STAT	XL UPSTRM	FLOWRATE	DELTA PRES	FLOW REGIMES OR REYNOLDS NO.
10	STUBE	1	2	1.0	1.0	NORM	1.000	0.	0.	0.
20	LOSS	2	3	3.0	1.0	NORM	1.000	0.	323.4	

MODEL = BALLOON FLUINT SAMPLE PROBLEM 1 - FILLING AIR BALLOONS
FORWRD

FLUID SUBMODEL NAME = FILLIT ; FLUID NO. = 8000

MAX TIME STEP = 5.591652E-05 ; LIMITING TANK = 3 REASON = COMPL. VOLUME CHANGE LIMIT
LAST TIME STEP = 4.628307E-05 VS. DTMAXF/DTMINF = 1.000000E+30 / 0. ; AVERAGE TIME STEP = 1.839173E-05
PROBLEM TIME TIMEN = 1.838806E-03 VS. TIMEND = 1.838806E-03

LUMP PARAMETER TABULATION FOR SUBMODEL FILLIT

LUMP	TYPE	TEMP	PRESSURE	QUALITY	DENSITY	ENTHALPY	HEAT RATE	MASS RATE	ENERGY RATE
1	TANK	60.11	301.9	1.000	1.645	124.5	0.	-125.5	-1.5623E+04
3	TANK	64.98	0.3703	1.000	7.7596E-02	125.6	0.	41.84	5208.
2	JUNC	60.11	234.4	1.000	1.294	124.5	0.	0.	0.

MODEL = BALLOON FLUINT SAMPLE PROBLEM 1 - FILLING AIR BALLOONS
FORWRD

FLUID SUBMODEL NAME = FILLIT ; FLUID NO. = 8000

MAX TIME STEP = 5.591652E-05 ; LIMITING TANK = 3 REASON = COMPL. VOLUME CHANGE LIMIT
LAST TIME STEP = 4.628307E-05 VS. DTMAXF/DTMINF = 1.000000E+30 / 0. ; AVERAGE TIME STEP = 1.839173E-05
PROBLEM TIME TIMEN = 1.838806E-03 VS. TIMEND = 1.838806E-03

LUMP EXTRA PARAMETER TABULATION FOR SUBMODEL FILLIT
BODY FORCE ACCELERATION COMPONENTS (X,Y,Z): 0. , 0. , 0.

LUMP	TYPE	CX	CY	CZ	NO. PATHS	HOLDS	MASS	VOLUME	COMPLIANCE	VOLUME RATE
1	TANK	0.	0.	0.	1	NONE	4.936	3.000	0.	0.
3	TANK	0.	0.	0.	1	NONE	7.9374E-02	1.023	1098.	0.
2	JUNC	0.	0.	0.	2	NONE				

MODEL = BALLOON
FORWRD

FLUINT SAMPLE PROBLEM 1 - FILLING AIR BALLOONS

FLUID SUBMODEL NAME = FILLIT ; FLUID NO. = 8000

MAX TIME STEP = 5.591652E-05 ; LIMITING TANK = 3 REASON = COMPL. VOLUME CHANGE LIMIT
LAST TIME STEP = 4.628307E-05 VS. DTMAXF/DTMINF = 1.000000E+30 / 0. ; AVERAGE TIME STEP = 1.839173E-05
PROBLEM TIME TIMEN = 1.838806E-03 VS. TIMEND = 1.838806E-03

PATH PARAMETER TABULATION FOR SUBMODEL FILLIT

PATH	TYPE	LMP 1	LMP 2	DUP I	DUP J	STAT	XL UPSTRM	FLOWRATE	DELTA PRES	FLOW REGIMES OR REYNOLDS NO.	
10	STUBE	1	2	1.0	1.0	NORM	1.000	125.5	67.49	1.17873E+06	1.17873E+06
20	LOSS	2	3	3.0	1.0	NORM	1.000	41.84	234.0		

PROBLEM B: SIPHON IN A ROMAN AQUADUCT

When the Romans built aquaducts to supply European cities with continuous water, they used for the most part open channels with slight hydraulic gradients. Unfortunately, this meant having to build now-famous bridges to span small valleys. The height limit on these masonry bridges was approximately 50m. When a valley was deeper than this limit, the Roman engineers used an ingenious but less-famous alternate: an inverted (U shaped) siphon. In the more common siphon, the difference in gravitational head between the inlet and outlet of a duct is used to drive the flow over a higher intermediate section, requiring a starter pump. In the inverted version, the outlet is lower than the inlet, but the intermediate section is lower still. Thus, the Romans used huge siphons along the bottom of deep valleys instead of tall bridges.*

In order to contain the large pressures at the bottoms of the siphons, the ducts were made out of thick lead that was rolled into a pear-shaped pipe and hammered or soldered shut. To carry the necessary flowrates and because of manufacturing limits on the pipe size, siphons typically used several such pipes. At the bottoms of the valleys, a relatively small bridge was used to limit the maximum pressure of the water by carrying the pipes above the lowest point. The siphons began and ended with open channel headers that connected to the rest of the aquaduct. The specific numbers used in this example are composites based on information available for several siphons.

This problem was chosen to demonstrate the following FLUINT features:

1. the properties of liquid-filled tanks
2. the use of body force options
3. the differences between tanks and junctions
4. the differences between tubes and STUBE connectors
5. heat transfer options; ties to thermal nodes
6. user-defined liquids
7. non-circular cross-sections
8. parametric runs (internal restarts)
9. LINE and HX macros
10. Multiple fluid and thermal submodels
11. INCLUDE options
12. again, the flexibility of the code

---

* All information in this example comes from: "Siphons in Roman Aquaducts," A. Trevor Hodge, *Scientific American*, June 1985, pp. 114–119.

B.1 Problem Description

A 1.2 kilometer siphon has eight 26 cm O.D. pipes (see Figures B–1 and B–2). The difference in the free surface height of the headers is 9m, and the bridge at the bottom of the valley is 93m below the inlet header. The fluid temperature is initially at room temperature (20° C). The total flowrate of the aquaduct and the pressure profile is desired. The relevant characteristics are:

Siphon (Fig. B–1):	
Total piping length	1.2 km
Hydraulic gradient	9 m
Bridge length	400 m
Bridge depth	93 m
Number of pipes in parallel	8
Pipe X-Section (Fig. B–2):	
Inner diameter	18 cm
Perimeter	60.4 cm
Flow area	272 cm^2
Hydraulic diameter	18 cm
Lead wall:	
Wall thickness	4 cm
Cross-sectional area	295 cm^2
Density	11340 kg/m^3
Specific heat	129 J/kg-K
Thermal conductivity	35.3 W/m-K

Two transient events are analyzed. These events are chosen to demonstrate program usage and are therefore somewhat whimsical. In the first event, a sudden snow melt causes the inlet temperature to drop linearly with time to 1 °C over a 10 minute interval. The desired output is the transient thermal response of the pipes. The pipes are embedded in masonry, and hence are assumed to be adiabatic at the outside diameter.

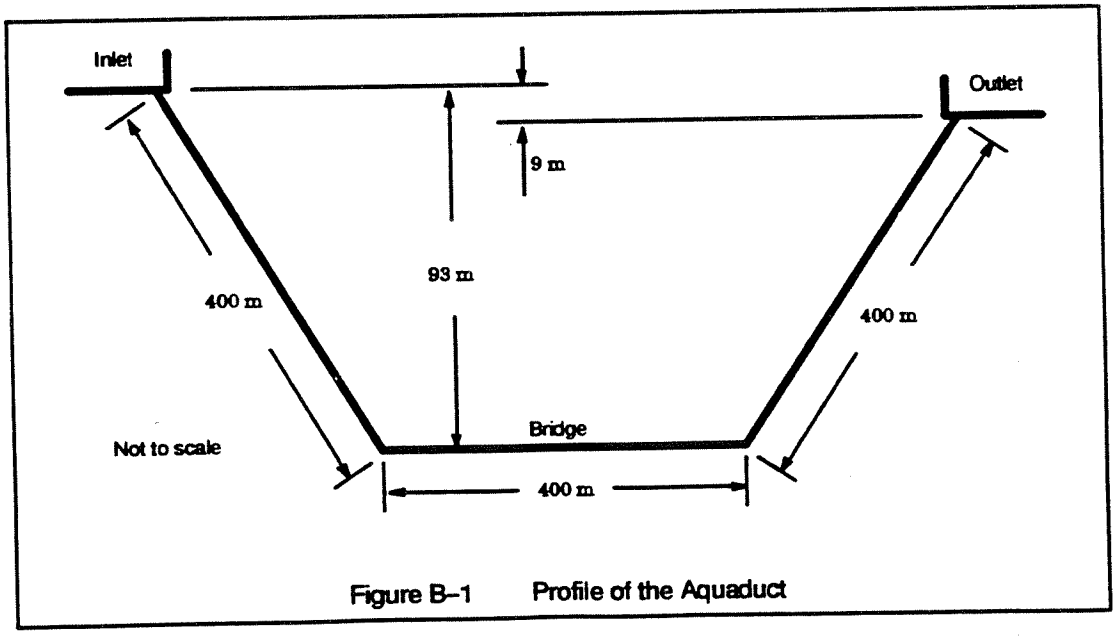


Figure B–1 Profile of the Aquaduct

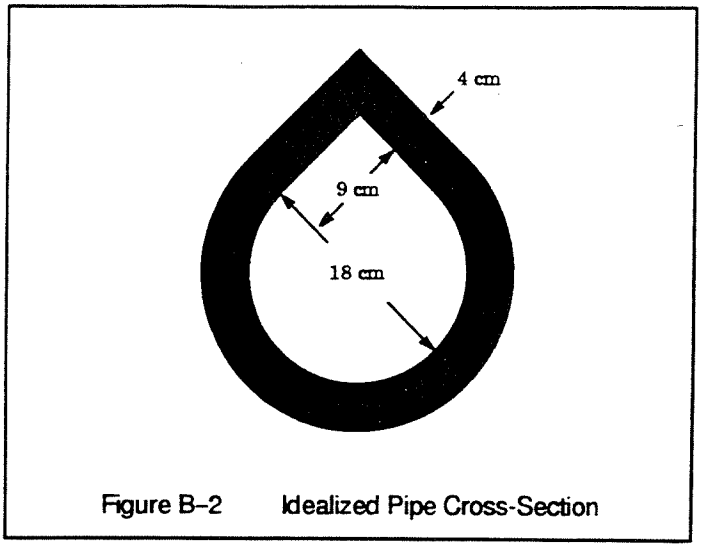


Figure B–2 Idealized Pipe Cross-Section

The second event uses the same initial conditions as the first event. In this case, however, an earthquake strikes at time zero, springing identical leaks in each pipe in the middle of the bridge. Each leak has a flow area of 10 cm^2, and the surrounding bridge is assumed to not inhibit the leakage flow. The desired output is the transient hydrodynamic response of the pipes. The effect of the flow changes on the liquid level in the headers is assumed negligible.

B.2 A FLUINT Model

Clearly, body forces will play an important role in this model*, and only one dimension is needed: the elevation from the bottom of the siphon. Also, flow losses need to be accurately modeled since the driving force is constant—the flowrate will depend on the losses in the pipes.

The modeling choice for the headers should be clear; the ducts will be bounded by two plena of different height but with the same initial temperature and pressure (atmospheric). The temperature of the inlet plenum can be changed using the CHGLMP routine for the snow melt transient. The leakages for the earthquake transient can be modeled as CTLVLVs that are closed until that event. These paths dump into a third plenum that is at the same height as the leak, but is still at atmospheric pressure. The FK for such a large leak can be taken as 1.0, the same as any exit.

* FLUINT currently neglects kinetic and potential energy terms; this analysis would have been invalid if the actual flowrates were limited by inviscid energy constraints and not by frictional losses.

In order to investigate modeling choices for the ducts themselves, each pipe will be modeled using a different combination. This will allow comparisons of the thermal and hydraulic transient response of each pipe model. The basic combinations will be as follows:

Pipe 1: Junctions and STUBE connectors, 29 segments, adiabatic
Pipe 2: Junctions and tubes, 29 segments, adiabatic
Pipe 3: Tanks and STUBE connectors, 29 segments, adiabatic
Pipe 4: Tanks and tubes, 29 segments, adiabatic
Pipe 5: Junctions and STUBE connectors, 29 segments, wall model included
Pipe 6: Tanks and tubes, 29 segments, wall model included
Pipe 7: Junctions and STUBE connectors, 15 segments, wall model included
Pipe 8: Tanks and tubes, 15 segments, wall model included

Note that these combinations will illustrate changes in spatial and temporal resolution. The input and the model is greatly complicated by this investigation; in a normal model the input and execution would be greatly simplified by using one duct model duplicated 8 times.

The wall roughness of the pipes can be expected to be very high considering the available manufacturing methods. A value of WRF=0.01 is assumed, meaning that the size of the roughness is on the order of 0.18 cm. (The pipes were constructed by joining sections that were 3m long; this roughness ratio will take into account frequent rough joints as well.)

With respect to heat transfer between the fluid and the wall for those diabatic ducts, one node is used per lump to represent the pipe. This is equivalent to assuming that the radial gradients in the pipe are negligible; this assumption can be lifted by adding more ties as needed. Axial gradients will be nonnegligible, and axial conductances will be added even though the axial heat transport along the pipe is negligible compared to the fluid transport. The real reason for axial conductors is to comply with the SINDA requirement that only boundary nodes can be isolated. Diffusion nodes will be used throughout to characterize the substantial thermal capacitance of the thick lead pipes. Also, although the pipes were probably heavily encrusted, they are assumed to have been recently cleaned in this example; no fouling resistance will be taken into account in the heat transfer model.

In order to fully demonstrate the LINE and HX macros, the first downhill segment will be downstream discretized, the center section will be center discretized, and the uphill section will be upstream discretized. This is displayed in Figure B–3 along with the naming conventions for all pipes. Because some pipes have fewer elements, they exclude the elements shown in parentheses. This does not mean that tank 10 for a 29 segment model corresponds exactly to tank 10 in a 15 segment model since the latter tank has twice the volume of the former.

Inlet and outlet losses need also be taken into account. An FK value of 0.5 is typical for a sharp inlet, while all exits have a value of 1.0 meaning that all velocity head is lost. The bends at either end of the bridge were slight but could be expected to be manufactured crudely; a value of 0.2 is attributed to each of them. These bend losses can be added into the entrance and exit losses for the purposes of all expected analyses. To interface with the duct models, an extra junction is inserted along with each LOSS connector. Alternatively (and more efficiently), these losses could have been superimposed on the entrance and exit paths of the ducts. In that case, the FK factors would be added in OPERATIONS DATA since specifying them within duct macros would apply them to all generated tubes or STUBE connectors.

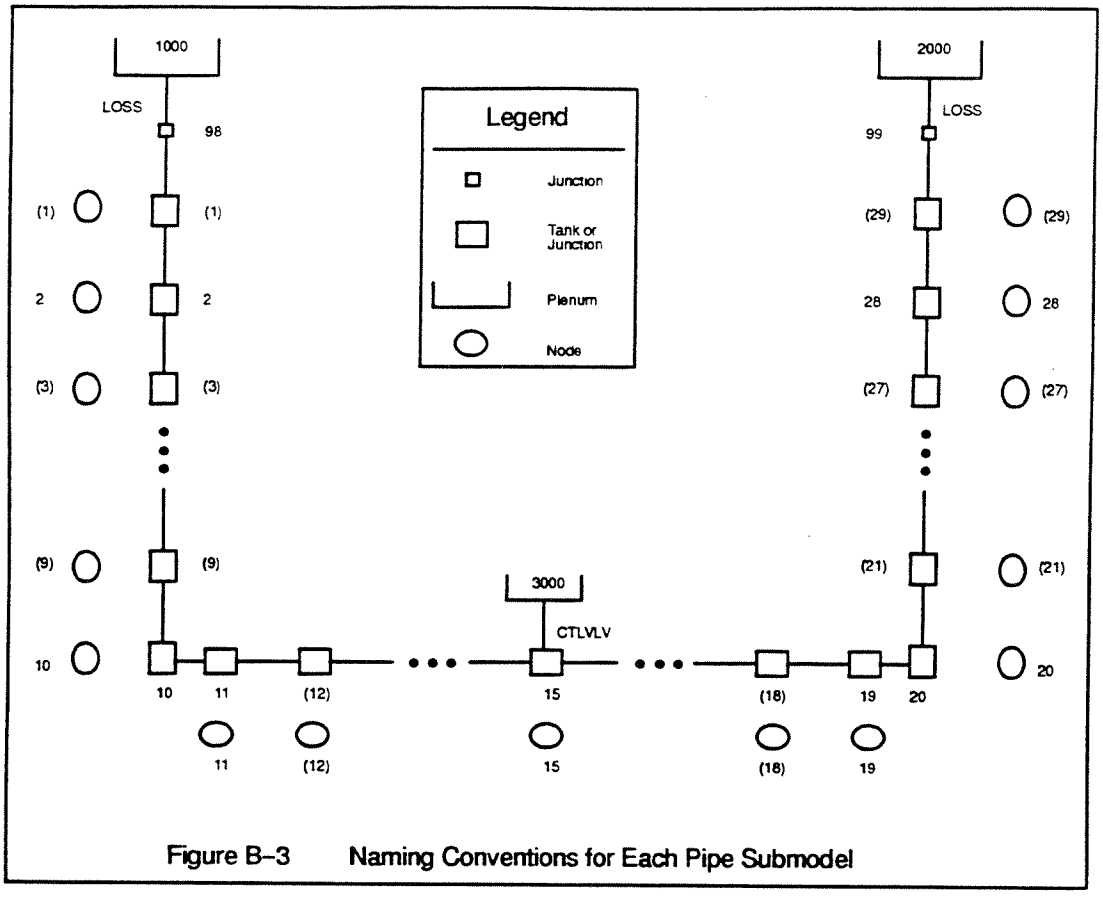


Figure B–3 Naming Conventions for Each Pipe Submodel

B.3 The Input File

The input file is listed immediately following the last section.

In order to demonstrate important submodel concepts, each pipe has been input as a separate fluid submodel, and each pipe wall has been input as a separate thermal submodel. Normally, a fluid system cannot be divided into submodels. However, in this case each pipe is completely independent of the other pipes because they all begin and end at plena. By subdividing the model into eight fluid submodels, intricate naming schemes can be avoided, and the same lump/path/node designators can be reused in each submodel in the corresponding location. The penalty that is paid is the need to repeat the common inputs describing the plena, the leaks, and the LOSS connectors and associated junctions. This work was minimized by writing this data once in a separate file, and then using the INCLUDE command to bring that data into each pipe submodel. The INCLUDE file contents are listed at the last page of the input file after the END OF DATA command.

Actually, there is a much more important but subtle reason to separate the pipes into submodels: the simulation will run much faster with 8 submodels having some 30 lumps each than will a model built with one 240-lump submodel. The reasons have to do with internal matrix operations whose cost scales approximately with the number of lumps squared, as discussed in the manual under Modeling Tips for Efficient Use. In other words, 8*(30**2) < 240**2. Therefore, although most fluid submodels cannot be subdivided, it is well worth the user's time to subdivide them where possible.

OPTIONS DATA—Like the previous balloon example, user files will be opened to contain summary tables of important data during each transient. The reader will note the use of these summary files throughout this appendix. SINDA/FLUINT output routines are necessarily general and complete, but a full dump of data can be cumbersome. Thus, a little simple Fortran programming allows the user to summarize the data that is really important. This is an important tool for scanning results to make sure that both the system and the model of that system are adequate. Another relevant tool is DATA_ONLY, which can be used to interrogate SAVE or RESAVE (RSO) files *after* an analysis has been completed, providing the user with the freedom to change his mind regarding which data are important enough to print or plot.

FPROP DATA—A full description of liquid water is provided in this section, and is named fluid 9718. It is more complete than the analysis warrants having been taking from the author's library of single-phase fluids. Deletion or simplification would speed the program, but is probably not worth the user's time. However, it is well worth the user's time to use an FPROP data block instead of relying on the complete internal description of water since the siphon is and will always be single-phase. This eliminates the substantial overhead involved in calculating liquid properties that are consistent with the vapor phase.

For analyses of fluid systems in which the working fluid is a standard FLUINT library fluid, the RAPPR program can be used to create customized single- and two-phase descriptions similar to the one provided above, which was instead taken from ASHRAE handbook data. This ASHRAE description will differ slightly from the RAPPR-produced description. Using RAPPR to generate a 9000 series (incompressible liquid) description of water allows the user to tailor the description to his accuracy requirements: more array points are used to comply with tighter error tolerance specifications. The reader is encouraged to repeat this analysis using a RAPPR description, but is cautioned against requesting very low error tolerances (less than 3%) because of the increased solution cost required by such a complete description.

NODE and CONDUCTOR DATA—These blocks contain the thermal network descriptions for the four thermal submodels used to represent the walls of pipes #5 through #8. They are named WALL5 through WALL8. GEN options are used to generate each segment of the duct corresponding to the LINE and HX macros in the relevant FLOW DATA blocks. Note that the calculations have been left to the program for documentation purposes. Actually, numbered user constants should have been used for the lead properties and the cross-sectional area. Changes in these factors would then be more controlled and better documented. The use of user constants was avoided to simplify the discussion. Note that user constants cannot be used as input in FLOW DATA blocks.

FLOW DATA—The eight submodel blocks corresponding to the eight pipe models are input here and named PIPE1 through PIPE8. Note that data common to all submodels is written once and INCLUDEd. The INCLUDE file, which includes the defaults, is listed at the end of the input list. All fluid submodels refer to the same working fluid (9718) that was described in the FPROP DATA block. The elevation of each lump with respect to the bottom is input as the Z coordinate.

Figure B–3 displays the naming conventions for each pipe submodel. The FLOW DATA sections contain three macros each for each leg of the aquaduct pipe. LINE macros are used to input nodeless models; HX macros are used to input models with ties to nodes. Note that the default values for LUINC, PAINC, and NINC are one, and that the diameter and flow area must be input explicitly as DHS and AFS. CZINC is used to increment the elevation along each duct. Each tie is named after the lump on which it operates.

CONTROL DATA—SI units are chosen, with temperature in units of °C, and pressure is relative to atmospheric. The acceleration of gravity in the Z direction is given in the appropriate units. (If the model had been in English units, the acceleration would have to be in units of ft/hr^2, or 32.174*3600.0*3600.0.) Also, the temperature changes in the thermal models have been limited by the constants EXTLIM and DTMPCA as preventative measures. EBALSA is set to zero to turn off the thermal energy balance check, which if active would encumber the convergence because there is no net energy flow through the thermal system at steady-state: the *relative* energy balance error would otherwise be compared against a diminishingly small *absolute* number.

USER DATA—The variables to be used in logic blocks are declared here. Note that the constant ITRAN is used to help the logic blocks distinguish between the snow melt transient (ITRAN=1) and the earthquake transient (ITRAN=2). Initially, ITRAN is zero.

OPERATIONS DATA—First, all submodels are built into one configuration. A steady-state analysis is run to find initial conditions such as the pressure profile along each pipe. Note that the call to STDSTL is actually redundant in this instance, but is it is normally a good practice to call this routine after calling FASTIC to make sure. The results of the steady-state run are saved for future use by calling SAVPAR, and pressures and flowrates are printed to the output file.

The snow melt transient is then run, with a large value of TIMEND being used because it will be overwritten in the FLOGIC 2 block. A "snapshot" of the end state after the snow melt transient is then taken, focusing on temperatures and heat rates. Note that the changing inlet plenum temperature is handled in the FLOGIC 0 block.

The earthquake (pipe break) transient is then run. First, the network steady-state solution is retrieved as initial conditions for the pipe break transient. Note that all control constant and variable changes occur *after* this retrieval; the control constants and variables themselves are stored and retrieved as part of the restart operation. The valves (leaks) are opened for all pipes before the run, and the pressures and flowrates are printed out after the run is completed.

FLOGIC 0—One FLOGIC 0 block is used to update values in all fluid submodels; alternatively, eight such blocks could have been input. The only operation required here is the update of the inlet temperatures during the snow melt transient. Note that the output interval OUTPTF assures that the program stops at exactly 600 seconds (10 minutes).

FLOGIC 2—Again, only one FLOGIC 2 block is used instead of eight. This logic alters the output interval depending on the transient event. During the pipe break transient, output is requested each time step instead of at regular intervals. The logic stops the simulation during the snow melt transient when the outlet temperature of PIPE6 falls below 5.0 °C, and it stops the pipe break transient when the network has relaxed enough to take a one second time step. Note that DTIMUF is the time step just taken; in FLOGIC 0 and 1, DTIMUF represents the time step taken in the *previous* solution step.

OUTPUT CALLS—This block writes one line of summary data for each event. Note that calls to generalized output routines are not made here; these are contained in the OPERATIONS DATA block to reduce the volume of output.

B.4 Output Description

The user summary files and selected parts of the OUTPUT file are listed following the input file. The results of each run are discussed separately.

STEADY-STATE—As should be the case, the flowrates are the same in each pipe: about 22.5 kg/s. Although no specific data is available for any one aquaduct, this flowrate is within 20% of values estimated for typical aquaducts. It may be a bit low, indicating that a roughness ratio of 0.01 may be too high; the Romans might be given more credit for smoother pipes, joints, and fittings.

The pressure profiles indicate a maximum pressure of 8.6 atmospheres. This is by no means the deepest siphon ever found; experiments indicate pipes built in modern times by ancient methods could withstand 18 atmospheres.

The pressure of the inlet junction (#98) is subatmospheric because it is at the same height as the inlet but experiences a loss due to connector #98. This is important to note because the model is valid even though no pressures are below atmospheric in a real siphon; the FLUINT model imitates the real system mathematically, not analogously. Junctions and connectors are artificial constructs useful as modeling tools; they do not necessarily have to correspond to a device on a schematic.

SNOW MELT EVENT—The temperature histories at the outlet of each pipe are plotted in Figure B–4. These profiles illustrate the important differences in modeling ducts.

First, note that there is no difference between using tubes and STUBE connectors. This event is a thermal transient that happens over the course of minutes and hours. Tubes are only needed to distinguish the more rapid hydraulic transients that happen over the course of seconds and microseconds. Thus, tubes are rarely needed in most thermal management systems.

Second, note that there is no lag time associated with junctions (pipes 1 and 2). Inlet changes are passed through instantaneously such that changes in the inlet plenum are reflected at the duct outlet immediately. This is obviously an inappropriate assumption for this event, but it is usually adequate in many real analyses and should be made whenever possible. Note that the inlet temperature was actually changed in a stair-step fashion rather than linearly, causing some artificial differences between the inlet and outlet temperatures.

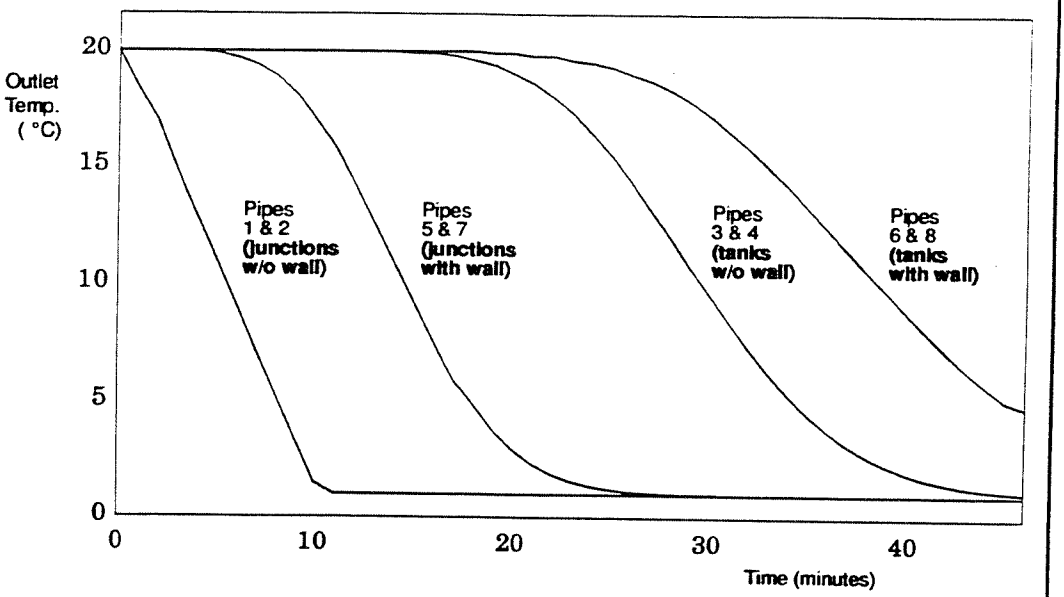


Figure B–4 Outlet Temperature Histories for Snow Melt Event

In models with tanks in place of junctions (pipes 3 and 4), the lag time associated with moving fluid is included, causing a delay between the inlet and the outlet. Note that the answers that result here differ slightly from the answers that would result if the fluid front were viewed as flat front that progresses at constant velocity through the length of the pipe. FLUINT uses a control volume approach instead. This means that the problem can be thought of as the transient cooling of the fluid in the middle of the pipe. *Neither assumption is completely correct.* FLUINT uses the control volume approach because of greater flexibility and modeling power; this approach has been shown to be acceptable in systems with temperature control valves where the lag effects are critical to the simulated response. Also, note that the ducts built with tanks do not respond for the first 20 to 25 minutes, which is equal to the amount of time required for the first cold front to reach the outlet.

When nodes are added to the model, the fluid is no longer considered adiabatic and the lag associated with transiently cooling the extensive amount of lead is included. The effect of thermal inertia on junctions can be seen in the response of pipes 5 and 7; the corresponding effect on tanks is seen in the response of pipes 6 and 8. Pipes 6 and 8 have the most accurate response since they contain the fewest simplifying assumptions. *This does not mean that tanks and diffusion nodes should always be used.* Simplifying assumptions should always be sought in any analyses.

Note that there is a negligibly small difference between the use of 29 lumps versus 15 (pipe 5 vs. 7, pipe 6 vs. 8). Clearly, acceptable answers can be found using coarse spatial resolution for liquids. Pipe 8 is then the best choice for this particular problem. However, the user should be warned that greater spatial resolution is required in two-phase ducts because of the large property gradients inherent in such systems.

EARTHQUAKE (PIPE BREAK) EVENT—Figure B–5 contains the outlet flowrate histories for the pipes during the pipe break event. The responses illustrate the differences between tubes and STUBE connectors. In those pipes built with tubes, the flowrate slows, reverses, and gradually approaches the steady-state value of -7.4 kg/s after about 15 seconds. The significant inertia of the moving liquid means that the leak starts slowly and builds to its full value as the flow is decelerated in the outlet and accelerated in the inlet. In those ducts built with STUBE connectors, the flowrates changes almost instantly to the final value. (There is some negligible delay: internal damping causes the flowrate to take two to three solution steps to arrive at the final answer.) Meanwhile, the inlet flowrate increases to 33 kg/s, and the leakage flowrate approaches 40 kg/s, the sum of the incoming flow.

The responses of the pipes are independent of whether they contain tanks or junctions. This is true only for noncompliant single-phase liquid systems, where pressures can propagate instantly due to the assumption of incompressibility. If the tanks were compliant (COMP > 0.0) or if any vapor were present, the responses would have differed. Actually, the astute reader will notice that there is some difference between STUBE models during the first several time steps depending on whether or not they contain junctions or tanks. Again, this difference is a result of the use of COMP=0.0 with liquid tanks: a hidden pressure propagation (zero time step residual term) solution is performed for the tank models, giving the STUBE damping logic one extra solution step over the junction models. Such nuances are usually invisible and unimportant.

In examining the output file, note that the time step is controlled to attempt to keep the tube flowrate percent change to DTSIZF. An important point to notice is that the time step decreases to capture a percentage change in a smaller flowrate as the flowrate reverses. Although tubes with negligibly small flowrates (<1% of model average) are ignored, tubes should be avoided in modeling ducts with small or zero flowrates, or where the flowrate reverses often.

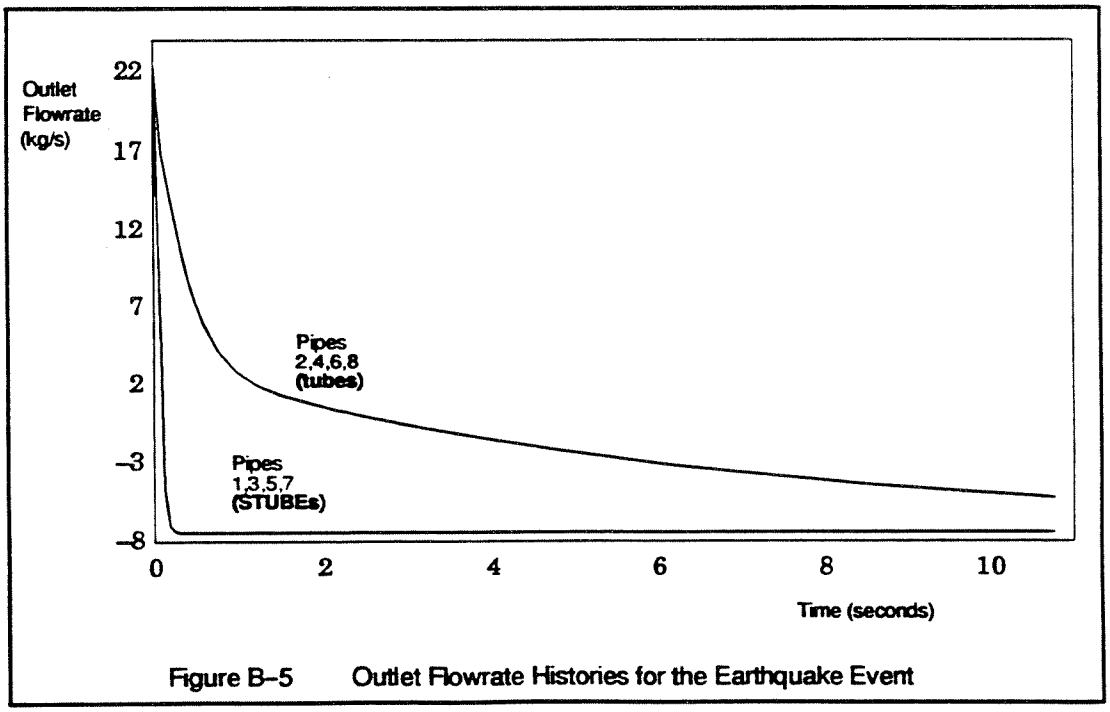


Figure B–5 Outlet Flowrate Histories for the Earthquake Event

As a point of interest, preliminary analyses found that the aquaduct would still transport 20 kg/s to the outlet even with a hole as large as 1 cm^2. Thus, the aquaducts would continue to function with minor leaks, which were probably common. The same is not true for the more familiar siphon having a raised center section.

Input File

```
HEADER OPTIONS DATA
TITLE FLUINT SAMPLE PROBLEM 2 - SIPHON IN A ROMAN AQUADUCT
       OUTPUT = aquaduct.out
       USER1  = snowmelt.out
       USER2  = earthquake.out
C
HEADER FPROP DATA,9718,ENG,-460.0
C
C MOST COMPLETE LIQUID WATER DESCRIPTION
C
       TMIN   = 32.0,TMAX   = 706.
AT,V,  32.0,4.28,     40.0,3.69,     50.0,3.12,     60.0,2.68,
       70.0,2.32,     80.0,2.03,     90.0,1.79,     100.0,1.6,
       120.0,1.30,    140.0,1.093,   160.0,0.938,   180.0,0.813,
       200.0,0.717,   220.0,0.638,   240.0,0.574,   260.0,0.521,
       280.0,0.476,   300.0,0.439,   320.0,0.406,   340.0,0.379,
       360.0,0.355,   400.0,0.315,   500.0,0.251,   600.0,0.200,
       700.0,0.121,   706.0,0.101
AT,K,  32.0,0.329,    50.0,0.339,    70.0,0.35,     90.0,0.359,
       120.0,0.371,   180.0,0.388,   220.0,0.395,   260.0,0.398
       280.0,0.398,   320.0,0.396,   360.0,0.391,   400.0,0.383
       500.0,0.349,   600.0,0.296,   700.0,0.188,   706.0,0.139
AT,CP, 32.0,1.007,    60.0,1.001,    90.0,0.998,    120.0,0.998,
       160.0,1.001,   180.0,1.003,   220.0,1.009,   240.0,1.013
       260.0,1.017,   280.0,1.022,   300.0,1.029,   320.0,1.040
       360.0,1.066,   400.0,1.085,   500.0,1.18,    600.0,1.528
AT,D,  33.0,1.0/0.01602,     50.0,1.0/0.01602
       60.0,1.0/0.01603,     80.0,1.0/0.01607
       120.0,1.0/0.01620,    170.0,1.0/0.01645
       190.0,1.0/0.01657,    212.0,1.0/0.01671,
       220.0,1.0/0.016772,   240.0,1.0/0.01692
       260.0,1.0/0.017084,   280.0,1.0/0.017259
       300.0,1.0/0.017448,   340.0,1.0/0.017872
       380.0,1.0/0.018363,   420.0,1.0/0.018936
       460.0,1.0/0.019614,   500.0,1.0/0.02043
C
```

```
HEADER NODE DATA,WALL5
C PIPE 5
   GEN 1,10,1,20.0,400.0*295.0E-4*11340.0*129.0/10.0      $ DOWN
   GEN 20,10,1,20.0,400.0*295.0E-4*11340.0*129.0/10.0     $ UP
   GEN 11,9,1,20.0,400.0*295.0E-4*11340.0*129.0/9.0       $ BRIDGE
C
HEADER NODE DATA,WALL6
C PIPE 6
   GEN 1,10,1,20.0,400.0*295.0E-4*11340.0*129.0/10.0      $ DOWN
   GEN 20,10,1,20.0,400.0*295.0E-4*11340.0*129.0/10.0     $ UP
   GEN 11,9,1,20.0,400.0*295.0E-4*11340.0*129.0/9.0       $ BRIDGE
C
HEADER NODE DATA,WALL7
C PIPE 7
   GEN 2,5,2,20.0,400.0*295.0E-4*11340.0*129.0/5.0        $ DOWN
   GEN 20,5,2,20.0,400.0*295.0E-4*11340.0*129.0/5.0       $ UP
   GEN 11,5,2,20.0,400.0*295.0E-4*11340.0*129.0/5.0       $ BRIDGE
C
HEADER NODE DATA,WALL8
C PIPE 8
   GEN 2,5,2,20.0,400.0*295.0E-4*11340.0*129.0/5.0        $ DOWN
   GEN 20,5,2,20.0,400.0*295.0E-4*11340.0*129.0/5.0       $ UP
   GEN 11,5,2,20.0,400.0*295.0E-4*11340.0*129.0/5.0       $ BRIDGE
C
```

```
HEADER CONDUCTOR DATA,WALL5
C PIPE 5
   GEN 1,9,1,1,1,2,1,35.3*295.0E-4*10.0/400.0          $ DOWN
   GEN 20,9,1,20,1,21,1,35.3*295.0E-4*10.0/400.0      $ UP
   GEN 11,8,1,11,1,12,1,35.3*295.0E-4*9.0/400.0       $ BRIDGE
   10,10,11,0.5*35.3*295.0E-4*9.0/400.0               $ BEND 1
   19,19,20,0.5*35.3*295.0E-4*9.0/400.0               $ BEND 2
C
HEADER CONDUCTOR DATA,WALL6
C PIPE 6
   GEN 1,9,1,1,1,2,1,35.3*295.0E-4*10.0/400.0          $ DOWN
   GEN 20,9,1,20,1,21,1,35.3*295.0E-4*10.0/400.0      $ UP
   GEN 11,8,1,11,1,12,1,35.3*295.0E-4*9.0/400.0       $ BRIDGE
   10,10,11,0.5*35.3*295.0E-4*9.0/400.0               $ BEND 1
   19,19,20,0.5*35.3*295.0E-4*9.0/400.0               $ BEND 2
C
HEADER CONDUCTOR DATA,WALL7
C PIPE 7
   GEN 2,4,2,2,2,4,2,35.3*295.0E-4*5.0/400.0          $ DOWN
   GEN 20,4,2,20,2,22,2,35.3*295.0E-4*5.0/400.0       $ UP
   GEN 11,4,2,11,2,13,2,35.3*295.0E-4*5.0/400.0       $ BRIDGE
   10,10,11,0.5*35.3*295.0E-4*5.0/400.0               $ BEND 1
   19,19,20,0.5*35.3*295.0E-4*5.0/400.0               $ BEND 2
C
HEADER CONDUCTOR DATA,WALL8
C PIPE 8
   GEN 2,4,2,2,2,4,2,35.3*295.0E-4*5.0/400.0          $ DOWN
   GEN 20,4,2,20,2,22,2,35.3*295.0E-4*5.0/400.0       $ UP
   GEN 11,4,2,11,2,13,2,35.3*295.0E-4*5.0/400.0       $ BRIDGE
   10,10,11,0.5*35.3*295.0E-4*5.0/400.0               $ BEND 1
   19,19,20,0.5*35.3*295.0E-4*5.0/400.0               $ BEND 2
C
```

```
HEADER FLOW DATA,PIPE1,FID=9718              $ START DESCR OF PIPE1
C
C INCLUDE BASIC INFO (SAME FOR ALL PIPES)
INCLUDE aquaduct.inc
C
C DOWNHILL LINE
M LINE,1,D,1,1,98,    NSEG = 10,     UPF = 0.0
       LU = JUNC,     PA = STUBE,    CZ = 0.9*93.0
       CZINC = -9.3,  TLENT = 400.0, DHS = 0.18,    AFS = 272.0E-4
C ACROSS BRIDGE
M LINE,2,C,11,11,10,20,       NSEG = 9,      UPF = 0.5
       LU = JUNC,     PA = STUBE,    CZ = 0.0
       CZINC = 0.0,   TLENT = 400.0, DHS = 0.18,    AFS = 272.0E-4
C UPHILL LINE
M LINE,3,U,20,21,99,  NSEG = 10,     UPF = 1.0
       LU = JUNC,     PA = STUBE,    CZ = 0.0
       CZINC = 8.4,   TLENT = 400.0, DHS = 0.18,    AFS = 272.0E-4
C
HEADER FLOW DATA,PIPE2,FID=9718              $ START DESCR OF PIPE2
C
C INCLUDE BASIC INFO (SAME FOR ALL PIPES)
INCLUDE aquaduct.inc
C
C DOWNHILL LINE
M LINE,1,D,1,1,98,    NSEG = 10,     UPF = 0.0
       LU = JUNC,     PA = TUBE,     CZ = 0.9*93.0
       CZINC = -9.3,  TLENT = 400.0, DHS = 0.18,    AFS = 272.0E-4
C ACROSS BRIDGE
M LINE,2,C,11,11,10,20,       NSEG = 9,      UPF = 0.5
       LU = JUNC,     PA = TUBE,     CZ = 0.0
       CZINC = 0.0,   TLENT = 400.0, DHS = 0.18,    AFS = 272.0E-4
C UPHILL LINE
M LINE,3,U,20,21,99,  NSEG = 10,     UPF = 1.0
       LU = JUNC,     PA = TUBE,     CZ = 0.0
       CZINC = 8.4,   TLENT = 400.0, DHS = 0.18,    AFS = 272.0E-4
C
```

```
HEADER FLOW DATA,PIPE3,FID=9718                $ START DESCR OF PIPE3
C
C INCLUDE BASIC INFO (SAME FOR ALL PIPES)
INCLUDE aquaduct.inc
C
C DOWNHILL LINE
M LINE,1,D,1,1,98,     NSEG = 10,      UPF = 0.0
       LU = TANK,      PA = STUBE,     CZ = 0.9*93.0
       CZINC = -9.3,   TLENT = 400.0, DHS = 0.18,    AFS = 272.0E-4
C ACROSS BRIDGE
M LINE,2,C,11,11,10,20,       NSEG = 9,       UPF = 0.5
       LU = TANK,      PA = STUBE,     CZ = 0.0
       CZINC = 0.0,    TLENT = 400.0, DHS = 0.18,    AFS = 272.0E-4
C UPHILL LINE
M LINE,3,U,20,21,99,   NSEG = 10,      UPF = 1.0
       LU = TANK,      PA = STUBE,     CZ = 0.0
       CZINC = 8.4,    TLENT = 400.0, DHS = 0.18,    AFS = 272.0E-4
C
HEADER FLOW DATA,PIPE4,FID=9718                $ START DESCR OF PIPE4
C
C INCLUDE BASIC INFO (SAME FOR ALL PIPES)
INCLUDE aquaduct.inc
C
C DOWNHILL LINE
M LINE,1,D,1,1,98,     NSEG = 10,      UPF = 0.0
       LU = TANK,      PA = TUBE,      CZ = 0.9*93.0
       CZINC = -9.3,   TLENT = 400.0, DHS = 0.18,    AFS = 272.0E-4
C ACROSS BRIDGE
M LINE,2,C,11,11,10,20,       NSEG = 9,       UPF = 0.5
       LU = TANK,      PA = TUBE,      CZ = 0.0
       CZINC = 0.0,    TLENT = 400.0, DHS = 0.18,    AFS = 272.0E-4
C UPHILL LINE
M LINE,3,U,20,21,99,   NSEG = 10,      UPF = 1.0
       LU = TANK,      PA = TUBE,      CZ = 0.0
       CZINC = 8.4,    TLENT = 400.0, DHS = 0.18,    AFS = 272.0E-4
C
```

```
HEADER FLOW DATA,PIPE5,FID=9718                $ START DESCR OF PIPE5
C
C INCLUDE BASIC INFO (SAME FOR ALL PIPES)
INCLUDE aquaduct.inc
C
C DOWNHILL LINE
M HX,1,D,1,1,1,WALL5.1,98,    NSEG = 10,      UPF = 0.0
       LU = JUNC,      PA = STUBE,     CZ = 0.9*93.0
       CZINC = -9.3,  TLENT = 400.0, DHS = 0.18,     AFS = 272.0E-4
C ACROSS BRIDGE
M HX,2,C,11,11,11,WALL5.11,10,20,    NSEG = 9,       UPF = 0.5
       LU = JUNC,      PA = STUBE,     CZ = 0.0
       CZINC = 0.0,   TLENT = 400.0, DHS = 0.18,     AFS = 272.0E-4
C UPHILL LINE
M HX,3,U,20,21,20,WALL5.20,99,       NSEG = 10,      UPF = 1.0
       LU = JUNC,      PA = STUBE,     CZ = 0.0
       CZINC = 8.4,   TLENT = 400.0, DHS = 0.18,     AFS = 272.0E-4
C
HEADER FLOW DATA,PIPE6,FID=9718                $ START DESCR OF PIPE6
C
C INCLUDE BASIC INFO (SAME FOR ALL PIPES)
INCLUDE aquaduct.inc
C
C DOWNHILL LINE
M HX,1,D,1,1,1,WALL6.1,98,    NSEG = 10,      UPF = 0.0
       LU = TANK,      PA = TUBE,      CZ = 0.9*93.0
       CZINC = -9.3,  TLENT = 400.0, DHS = 0.18,     AFS = 272.0E-4
C ACROSS BRIDGE
M HX,2,C,11,11,11,WALL6.11,10,20,    NSEG = 9,       UPF = 0.5
       LU = TANK,      PA = TUBE,      CZ = 0.0
       CZINC = 0.0,   TLENT = 400.0, DHS = 0.18,     AFS = 272.0E-4
C UPHILL LINE
M HX,3,U,20,21,20,WALL6.20,99,       NSEG = 10,      UPF = 1.0
       LU = TANK,      PA = TUBE,      CZ = 0.0
       CZINC = 8.4,   TLENT = 400.0, DHS = 0.18,     AFS = 272.0E-4
C
```

```
HEADER FLOW DATA,PIPE7,FID=9718                $ START DESCR OF PIPE7
C
C INCLUDE BASIC INFO (SAME FOR ALL PIPES)
INCLUDE aquaduct.inc
C
C DOWNHILL LINE
M HX,1,D,2,2,2,WALL7.2,98,   NSEG = 5,      UPF = 0.0
       LU = JUNC,     PA = STUBE,    CZ = 0.8*93.0
       LUINC = 2,     PAINC = 2,     NINC = 2,      TIINC  = 2
       CZINC = -18.6, TLENT = 400.0, DHS = 0.18,    AFS = 272.0E-4
C ACROSS BRIDGE
M HX,2,C,11,11,11,WALL7.11,10,20,   NSEG = 5,      UPF = 0.5
       LU = JUNC,     PA = STUBE,    CZ = 0.0
       LUINC = 2,     PAINC = 2,     NINC = 2,      TIINC  = 2
       CZINC = 0.0,   TLENT = 400.0, DHS = 0.18,    AFS = 272.0E-4
C UPHILL LINE
M HX,3,U,20,22,20,WALL7.20,99,      NSEG = 5,      UPF = 1.0
       LU = JUNC,     PA = STUBE,    CZ = 0.0
       LUINC = 2,     PAINC = 2,     NINC = 2,      TIINC  = 2
       CZINC = 16.8,  TLENT = 400.0, DHS = 0.18,    AFS = 272.0E-4
C
HEADER FLOW DATA,PIPE8,FID=9718                $ START DESCR OF PIPE8
C
C INCLUDE BASIC INFO (SAME FOR ALL PIPES)
INCLUDE aquaduct.inc
C
C DOWNHILL LINE
M HX,1,D,2,2,2,WALL8.2,98,   NSEG = 5,      UPF = 0.0
       LU = TANK,     PA = TUBE,     CZ = 0.8*93.0
       LUINC = 2,     PAINC = 2,     NINC = 2,      TIINC  = 2
       CZINC = -18.6, TLENT = 400.0, DHS = 0.18,    AFS = 272.0E-4
C ACROSS BRIDGE
M HX,2,C,11,11,11,WALL8.11,10,20,   NSEG = 5,      UPF = 0.5
       LU = TANK,     PA = TUBE,     CZ = 0.0
       LUINC = 2,     PAINC = 2,     NINC = 2,      TIINC  = 2
       CZINC = 0.0,   TLENT = 400.0, DHS = 0.18,    AFS = 272.0E-4
C UPHILL LINE
M HX,3,U,20,22,20,WALL8.20,99,      NSEG = 5,      UPF = 1.0
       LU = TANK,     PA = TUBE,     CZ = 0.0
       LUINC = 2,     PAINC = 2,     NINC = 2,      TIINC  = 2
       CZINC = 16.8,  TLENT = 400.0, DHS = 0.18,    AFS = 272.0E-4
C
```

```
HEADER CONTROL DATA,GLOBAL
       NLOOPS = 100                 $ MAX OF 100 SS ITERATIONS
       OUTPUT = 1000.0              $ LARGE FAKE OUTPUT INTERVAL
       OUTPTF = 1000.0              $ LARGE FAKE OUTPUT INTERVAL
       EXTLIM = 5.0                 $ LIMIT THERMAL EXTRAPOLATIONS
       DTMPCA = 2.0                 $ LIMIT THERMAL CHANGES TO 10%
       EBALSA = 0.0                 $ NO NET ENERGY FLOW IN THERMAL
C
       UID    = SI                  $ METRIC UNITS
       ABSZRO = -273.16             $ DEGREES C
       PATMOS = -101325.0           $ GAUGE PRESSURE
C
       ACCELZ = 9.81                $ ACCEL OF GRAVITY
C
HEADER USER DATA,GLOBAL
       NREC   = 0                   $ PARAMETRIC RECORD NUMBER
       ITRAN  = 0                   $ RUN FLAG FOR EVENTS
       TINLET = 20.0                $ INFLOW TEMPERATURE
       I      = 0                   $ USED IN OPERATIONS DATA
C
```

```
HEADER OPERATIONS DATA
BUILD AQUA,WALL5,WALL6,WALL7,WALL8
BUILDF AQUA,PIPE1,PIPE2,PIPE3,PIPE4,PIPE5,PIPE6,PIPE7,PIPE8
C
       CALL FASTIC                    $ SMOOTH INIT COND
       CALL STDSTL                    $ MAKE SURE OF STEADY STATE
       CALL SAVPAR(NREC)              $ SAVE FOR FUTURE USE
C
C OUTPUT STEADY STATE FLOWRATES AND PRESSURES
C
       CALL LMPTAB('ALL')
       CALL PTHTAB('ALL')
C
C SET UP AND RUN SNOWMELT TRANSIENT
C
       WRITE(NUSER1,10)(I,I=1,8)
10      FORMAT(' SNOWMELT TRANSIENT SUMMARY: OUTLET TEMPERATURE'//
     .   '    TIME (SEC)',T15,8(4X,'PIPE',I2,2X)/)
C
       ITRAN          = 1
       TIMEND         = 10000.0      $ LARGE ARTIFICIAL TIMEND
       CALL FORWRD
C
C TAKE SNAPSHOT AT END OF SNOWMELT TRANSIENT
C
       CALL LMPTAB('ALL')
       CALL PTHTAB('ALL')
       CALL TIETAB('ALL')
       CALL TPRINT('ALL')
C
C SET UP AND RUN PIPE BREAK (EARTHQUAKE) TRANSIENT
C
       WRITE(NUSER2,20)(I,I=1,8)
20      FORMAT(' EARTHQUAKE TRANSIENT SUMMARY: OUTLET FLOWRATE'//
     .   '    TIME (SEC)',T15,8(4X,'PIPE',I2,2X)/)
C
       CALL RESPAR(NREC)              $ RETRIEVE STEADY STATE
       ITRAN          = 2             $ CHANGE AFTER RESPAR
       TIMEO          = 0.0           $ RESET TIME ZERO
       PIPE1.FK3000   = 1.0           $ OPEN LEAKS AT TIME ZERO
       PIPE2.FK3000   = 1.0
       PIPE3.FK3000   = 1.0
       PIPE4.FK3000   = 1.0
       PIPE5.FK3000   = 1.0
       PIPE6.FK3000   = 1.0
       PIPE7.FK3000   = 1.0
       PIPE8.FK3000   = 1.0
       TIMEND         = 100.0         $ LARGE ARTIFICIAL TIMEND
```

```
        CALL FORWRD
C
C TAKE SNAPSHOT AT END OF EARTHQUAKE TRANSIENT
C
        CALL LMPTAB('ALL')
        CALL PTHTAB('ALL')
DEFMOD PIPE1
        WRITE(NUSER2,30) FR98, FR3000
30      FORMAT(//' FINAL INLET FLOWRATE:',T30,1PG12.3/
     .          ' FINAL LEAKAGE FLOWRATE:',T30,G12.3)
C
HEADER FLOGIC 0,PIPE1                          $ LOGIC FOR ALL PIPES
C
C THE FOLLOWING LOGIC ADJUSTS THE INLET TEMPERATURE TO DECREASE
C LINEARLY WITH TIME FOR THE SNOWMELT TRANSIENT.
C
        IF(ITRAN .EQ. 1 .AND. TIMEN .LE. 610.0) THEN
                TINLET          = MAX(1.0,20.0-19.0*TIMEN/600.0)
                CALL CHGLMP('PIPE1',1000,'TL',TINLET,'PL')
                CALL CHGLMP('PIPE2',1000,'TL',TINLET,'PL')
                CALL CHGLMP('PIPE3',1000,'TL',TINLET,'PL')
                CALL CHGLMP('PIPE4',1000,'TL',TINLET,'PL')
                CALL CHGLMP('PIPE5',1000,'TL',TINLET,'PL')
                CALL CHGLMP('PIPE6',1000,'TL',TINLET,'PL')
                CALL CHGLMP('PIPE7',1000,'TL',TINLET,'PL')
                CALL CHGLMP('PIPE8',1000,'TL',TINLET,'PL')
        ENDIF
C
HEADER FLOGIC 2,PIPE1                          $ LOGIC FOR ALL PIPES
C
C THE FOLLOWING LOGIC SELECTS AN APPROPRIATE OUTPUT INTERVAL
C AND AN APPROPRIATE END TIME
C
        IF(ITRAN .EQ. 1)THEN
                OUTPTF          = 60.0
                IF(PIPE6.TL99 .LE. 5.0) TIMEND      = MAX(TIMEN,600.0)
        ELSEIF(ITRAN .EQ. 2)THEN
                OUTPTF          = 1000.0
                OPITRF          = 1.0          $ OUTPUT EACH STEP
                IF(PIPE6.DTIMUF .GT. 1.0) TIMEND = TIMEN
        ENDIF
C
```

```
HEADER OUTPUT CALLS, PIPE1
C
       IF(ITRAN .EQ. 1)THEN
              WRITE(NUSER1,10)TIMEN,TL99,PIPE2.TL99,PIPE3.TL99,
     .        PIPE4.TL99,PIPE5.TL99,PIPE6.TL99,PIPE7.TL99,PIPE8.TL99
       ELSEIF(ITRAN .EQ. 2)THEN
              WRITE(NUSER2,10)TIMEN,FR99,PIPE2.FR99,PIPE3.FR99,
     .        PIPE4.FR99,PIPE5.FR99,PIPE6.FR99,PIPE7.FR99,PIPE8.FR99
       ENDIF
10     FORMAT(1X,1PG13.6,8G12.3)
END OF DATA



C
C THIS IS THE INCLUDE FILE FOR ALL AQUADUCT PIPE SUBMODELS
C
LU DEF, TL = 20.0,     PL = 0.0,      XL = 0.0
PA DEF, FR = 25.0,     WRF = 0.01    $ ~20000 m3/day/aquaduct
       DH = 0.18,      AF = 272.0E-4
C
LU PLEN,1000,  CZ = 93.0                              $ INLET
LU PLEN,2000,  CZ = 84.0                              $ OUTLET
LU PLEN,3000,  CZ = 0.0                               $ EXIT
PA CONN,98,1000,98,   DEV = LOSS,    FK = 0.7         $ ENTRANCE, BEND
PA CONN,99,99,2000,   DEV = LOSS,    FK = 1.2         $ EXIT, BEND
LU JUNC,98,    CZ = 93.0                              $ ENTRANCE
LU JUNC,99,    CZ = 84.0                              $ EXIT
PA CONN,3000,15,3000, DEV = CTLVLV,  FK = -1.0        $ LEAK
              AF = 10.0E-4
```

Processor Output

SNOWMELT TRANSIENT SUMMARY: OUTLET TEMPERATURE

TIME (SEC)	PIPE 1	PIPE 2	PIPE 3	PIPE 4	PIPE 5	PIPE 6	PIPE 7	PIPE 8
0.	20.0	20.0	20.0	20.0	20.0	20.0	20.0	20.0
60.0000	18.6	18.6	20.0	20.0	20.0	20.0	20.0	20.0
120.000	16.7	16.7	20.0	20.0	20.0	20.0	20.0	20.0
180.000	14.8	14.8	20.0	20.0	20.0	20.0	20.0	20.0
240.000	12.9	12.9	20.0	20.0	20.0	20.0	20.0	20.0
300.000	11.0	11.0	20.0	20.0	19.9	20.0	19.9	20.0
360.000	9.11	9.11	20.0	20.0	19.8	20.0	19.6	20.0
420.000	7.21	7.21	20.0	20.0	19.4	20.0	19.2	20.0
480.000	5.31	5.31	20.0	20.0	18.9	20.0	18.6	20.0
540.000	3.41	3.41	20.0	20.0	18.1	20.0	17.8	20.0
600.000	1.51	1.51	20.0	20.0	17.0	20.0	16.7	20.0
660.000	1.02	1.02	20.0	20.0	15.9	20.0	15.6	20.0
720.000	1.02	1.02	20.0	20.0	14.2	20.0	14.0	20.0
780.000	1.02	1.02	20.0	20.0	12.6	20.0	12.4	20.0
840.000	1.02	1.02	20.0	20.0	10.8	20.0	10.7	20.0
900.000	1.02	1.02	20.0	20.0	9.13	20.0	9.09	20.0
960.000	1.02	1.02	19.9	19.9	7.51	20.0	7.57	20.0
1020.00	1.02	1.02	19.8	19.8	6.17	20.0	6.30	19.9
1080.00	1.02	1.02	19.6	19.6	4.80	20.0	5.00	19.9
1140.00	1.02	1.02	19.4	19.4	3.77	20.0	4.01	19.8
1200.00	1.02	1.02	19.1	19.1	2.96	19.9	3.21	19.7
1260.00	1.02	1.02	18.6	18.6	2.35	19.9	2.59	19.5
1320.00	1.02	1.02	18.0	18.0	1.90	19.8	2.12	19.4
1380.00	1.02	1.02	17.3	17.3	1.59	19.7	1.78	19.2
1440.00	1.02	1.02	16.4	16.5	1.38	19.5	1.53	18.9
1500.00	1.02	1.02	15.5	15.5	1.24	19.3	1.36	18.6
1560.00	1.02	1.02	14.4	14.4	1.15	19.0	1.24	18.2
1620.00	1.02	1.02	13.2	13.2	1.10	18.7	1.16	17.8
1680.00	1.02	1.02	12.0	12.0	1.07	18.3	1.11	17.3
1740.00	1.02	1.02	10.8	10.8	1.05	17.8	1.08	16.7
1800.00	1.02	1.02	9.55	9.55	1.04	17.2	1.06	16.1
1860.00	1.02	1.02	8.38	8.38	1.03	16.6	1.04	15.5
1920.00	1.02	1.02	7.27	7.28	1.03	15.8	1.04	14.8
1980.00	1.02	1.02	6.25	6.26	1.02	15.1	1.03	14.1
2040.00	1.02	1.02	5.32	5.33	1.02	14.2	1.03	13.4
2100.00	1.02	1.02	4.52	4.52	1.02	13.4	1.02	12.6
2160.00	1.02	1.02	3.82	3.82	1.02	12.4	1.02	11.9
2220.00	1.02	1.02	3.23	3.23	1.02	11.5	1.02	11.1
2280.00	1.02	1.02	2.74	2.74	1.02	10.6	1.02	10.4
2340.00	1.02	1.02	2.34	2.34	1.02	9.71	1.02	9.62
2400.00	1.02	1.02	2.02	2.02	1.02	8.84	1.02	8.91
2460.00	1.02	1.02	1.76	1.77	1.02	8.01	1.02	8.23
2520.00	1.02	1.02	1.57	1.57	1.02	7.21	1.02	7.57
2580.00	1.02	1.02	1.42	1.42	1.02	6.47	1.02	6.95
2640.00	1.02	1.02	1.31	1.31	1.02	5.78	1.02	6.36
2700.00	1.02	1.02	1.23	1.23	1.02	5.15	1.02	5.81
2730.00	1.02	1.02	1.19	1.19	1.02	4.85	1.02	5.55

EARTHQUAKE TRANSIENT SUMMARY: OUTLET FLOWRATE

TIME (SEC)	PIPE 1	PIPE 2	PIPE 3	PIPE 4	PIPE 5	PIPE 6	PIPE 7	PIPE 8
0.	22.4	22.4	22.4	22.4	22.4	22.4	22.4	22.4
2.771594E-02	2.61	20.5	-7.99	20.5	2.61	20.5	2.61	20.5
5.661675E-02	-1.31	19.2	-8.12	19.2	-1.31	19.2	-2.49	19.2
9.933728E-02	-4.75	17.7	-7.89	17.7	-4.75	17.7	-5.57	17.7
0.151752	-6.69	16.0	-7.68	16.0	-6.69	16.0	-6.86	16.0
0.200547	-7.22	14.5	-7.54	14.5	-7.22	14.5	-7.21	14.5
0.246559	-7.33	13.1	-7.48	13.1	-7.32	13.1	-7.33	13.1
0.290642	-7.39	11.9	-7.45	11.9	-7.39	11.9	-7.40	11.9
0.333408	-7.42	10.8	-7.43	10.8	-7.42	10.8	-7.42	10.8
0.375471	-7.43	9.80	-7.43	9.80	-7.43	9.80	-7.43	9.80
0.417187	-7.43	8.91	-7.43	8.91	-7.43	8.91	-7.43	8.91
0.458886	-7.43	8.11	-7.43	8.11	-7.43	8.11	-7.43	8.11
0.500846	-7.43	7.38	-7.43	7.38	-7.43	7.38	-7.43	7.38
0.543210	-7.43	6.71	-7.43	6.71	-7.43	6.71	-7.43	6.71
0.586151	-7.43	6.11	-7.43	6.11	-7.43	6.11	-7.43	6.11
0.629860	-7.43	5.57	-7.43	5.57	-7.43	5.57	-7.43	5.57
0.674533	-7.43	5.07	-7.43	5.07	-7.43	5.07	-7.43	5.07
0.720374	-7.43	4.62	-7.43	4.62	-7.43	4.62	-7.43	4.62
0.767593	-7.43	4.22	-7.43	4.22	-7.43	4.22	-7.43	4.22
0.816405	-7.43	3.85	-7.43	3.85	-7.43	3.85	-7.43	3.85
0.867031	-7.43	3.51	-7.43	3.51	-7.43	3.51	-7.43	3.51
0.919691	-7.43	3.20	-7.43	3.20	-7.43	3.20	-7.43	3.20
0.974591	-7.43	2.92	-7.43	2.92	-7.43	2.92	-7.43	2.92
1.03191	-7.43	2.67	-7.43	2.67	-7.43	2.67	-7.43	2.67
1.09177	-7.43	2.43	-7.43	2.43	-7.43	2.43	-7.43	2.43
1.15423	-7.43	2.22	-7.43	2.22	-7.43	2.22	-7.43	2.22
1.21920	-7.43	2.02	-7.43	2.02	-7.43	2.02	-7.43	2.02
1.28646	-7.43	1.84	-7.43	1.84	-7.43	1.84	-7.43	1.84
1.35560	-7.43	1.68	-7.43	1.68	-7.43	1.68	-7.43	1.68
1.42603	-7.43	1.53	-7.43	1.53	-7.43	1.53	-7.43	1.53
1.49701	-7.43	1.39	-7.43	1.39	-7.43	1.39	-7.43	1.39
1.56766	-7.43	1.26	-7.43	1.26	-7.43	1.26	-7.43	1.26
1.63712	-7.43	1.14	-7.43	1.14	-7.43	1.14	-7.43	1.14
1.70458	-7.43	1.04	-7.43	1.04	-7.43	1.04	-7.43	1.04
1.76933	-7.43	0.937	-7.43	0.937	-7.43	0.937	-7.43	0.937
1.83086	-7.43	0.847	-7.43	0.847	-7.43	0.847	-7.43	0.847
1.88881	-7.43	0.765	-7.43	0.765	-7.43	0.765	-7.43	0.765
1.94298	-7.43	0.690	-7.43	0.690	-7.43	0.690	-7.43	0.690
1.99330	-7.43	0.623	-7.43	0.623	-7.43	0.623	-7.43	0.623
2.03980	-7.43	0.562	-7.43	0.562	-7.43	0.562	-7.43	0.562
2.08260	-7.43	0.507	-7.43	0.507	-7.43	0.507	-7.43	0.507
2.12186	-7.43	0.457	-7.43	0.457	-7.43	0.457	-7.43	0.457
2.15776	-7.43	0.412	-7.43	0.412	-7.43	0.412	-7.43	0.412
2.19052	-7.43	0.371	-7.43	0.371	-7.43	0.371	-7.43	0.371
2.22036	-7.43	0.334	-7.43	0.334	-7.43	0.334	-7.43	0.334
2.24749	-7.43	0.301	-7.43	0.301	-7.43	0.301	-7.43	0.301
2.27213	-7.43	0.271	-7.43	0.271	-7.43	0.271	-7.43	0.271
2.29448	-7.43	0.244	-7.43	0.244	-7.43	0.244	-7.43	0.244
2.31472	-7.43	0.220	-7.43	0.220	-7.43	0.220	-7.43	0.220
2.33306	-7.43	0.198	-7.43	0.198	-7.43	0.198	-7.43	0.198
2.35041	-7.43	0.177	-7.43	0.177	-7.43	0.177	-7.43	0.177
2.36783	-7.43	0.157	-7.43	0.157	-7.43	0.157	-7.43	0.157
2.38531	-7.43	0.136	-7.43	0.136	-7.43	0.136	-7.43	0.136
2.40284	-7.43	0.116	-7.43	0.116	-7.43	0.116	-7.43	0.116
2.42044	-7.43	9.502E-02	-7.43	9.503E-02	-7.43	9.502E-02	-7.43	9.502E-02
2.43808	-7.43	7.447E-02	-7.43	7.447E-02	-7.43	7.447E-02	-7.43	7.446E-02
2.45578	-7.43	5.394E-02	-7.43	5.394E-02	-7.43	5.394E-02	-7.43	5.393E-02
2.47313	-7.43	3.389E-02	-7.43	3.389E-02	-7.43	3.389E-02	-7.43	3.388E-02
2.48789	-7.43	1.688E-02	-7.43	1.688E-02	-7.43	1.688E-02	-7.43	1.687E-02
2.49839	-7.43	4.805E-03	-7.43	4.809E-03	-7.43	4.806E-03	-7.43	4.800E-03
2.50267	-7.43	-1.077E-04	-7.43	-1.039E-04	-7.43	-1.065E-04	-7.43	-1.124E-04
2.51122	-7.43	-9.917E-03	-7.43	-9.912E-03	-7.43	-9.914E-03	-7.43	-9.920E-03
2.52833	-7.43	-2.947E-02	-7.43	-2.946E-02	-7.43	-2.946E-02	-7.43	-2.947E-02
2.54631	-7.43	-4.994E-02	-7.43	-4.993E-02	-7.43	-4.993E-02	-7.43	-4.994E-02
2.56436	-7.43	-7.042E-02	-7.43	-7.040E-02	-7.43	-7.041E-02	-7.43	-7.041E-02
2.58247	-7.43	-9.090E-02	-7.43	-9.088E-02	-7.43	-9.088E-02	-7.43	-9.089E-02
2.60065	-7.43	-0.111	-7.43	-0.111	-7.43	-0.111	-7.43	-0.111
2.61889	-7.43	-0.132	-7.43	-0.132	-7.43	-0.132	-7.43	-0.132
2.63720	-7.43	-0.152	-7.43	-0.152	-7.43	-0.152	-7.43	-0.152
2.65557	-7.43	-0.173	-7.43	-0.173	-7.43	-0.173	-7.43	-0.173
2.67400	-7.43	-0.193	-7.43	-0.193	-7.43	-0.193	-7.43	-0.193
2.69250	-7.43	-0.214	-7.43	-0.214	-7.43	-0.214	-7.43	-0.214
2.71179	-7.43	-0.235	-7.43	-0.235	-7.43	-0.235	-7.43	-0.235
2.73309	-7.43	-0.259	-7.43	-0.259	-7.43	-0.259	-7.43	-0.259
2.75659	-7.43	-0.284	-7.43	-0.284	-7.43	-0.284	-7.43	-0.284
2.78254	-7.43	-0.313	-7.43	-0.313	-7.43	-0.313	-7.43	-0.313
2.81120	-7.43	-0.344	-7.43	-0.344	-7.43	-0.344	-7.43	-0.344
2.84286	-7.43	-0.378	-7.43	-0.378	-7.43	-0.378	-7.43	-0.378
2.87786	-7.43	-0.415	-7.43	-0.415	-7.43	-0.415	-7.43	-0.415
2.91657	-7.43	-0.457	-7.43	-0.457	-7.43	-0.457	-7.43	-0.457
2.95939	-7.43	-0.502	-7.43	-0.502	-7.43	-0.502	-7.43	-0.502
3.00680	-7.43	-0.552	-7.43	-0.552	-7.43	-0.552	-7.43	-0.552
3.05932	-7.43	-0.607	-7.43	-0.607	-7.43	-0.607	-7.43	-0.607
3.11752	-7.43	-0.667	-7.43	-0.667	-7.43	-0.667	-7.43	-0.667
3.18208	-7.43	-0.733	-7.43	-0.733	-7.43	-0.733	-7.43	-0.733
3.25375	-7.43	-0.805	-7.43	-0.805	-7.43	-0.805	-7.43	-0.805
3.33338	-7.43	-0.885	-7.43	-0.885	-7.43	-0.885	-7.43	-0.885
3.42194	-7.43	-0.972	-7.43	-0.972	-7.43	-0.972	-7.43	-0.972
3.52057	-7.43	-1.07	-7.43	-1.07	-7.43	-1.07	-7.43	-1.07
3.63054	-7.43	-1.17	-7.43	-1.17	-7.43	-1.17	-7.43	-1.17
3.75335	-7.43	-1.29	-7.43	-1.29	-7.43	-1.29	-7.43	-1.29
3.89072	-7.43	-1.41	-7.43	-1.41	-7.43	-1.41	-7.43	-1.41
4.04469	-7.43	-1.55	-7.43	-1.55	-7.43	-1.55	-7.43	-1.55
4.21763	-7.43	-1.70	-7.43	-1.70	-7.43	-1.70	-7.43	-1.70
4.41239	-7.43	-1.87	-7.43	-1.87	-7.43	-1.87	-7.43	-1.87
4.63236	-7.43	-2.05	-7.43	-2.05	-7.43	-2.05	-7.43	-2.05
4.88163	-7.43	-2.25	-7.43	-2.25	-7.43	-2.25	-7.43	-2.25
5.16522	-7.43	-2.46	-7.43	-2.46	-7.43	-2.46	-7.43	-2.46
5.48933	-7.43	-2.70	-7.43	-2.70	-7.43	-2.70	-7.43	-2.70
5.86174	-7.43	-2.95	-7.43	-2.95	-7.43	-2.95	-7.43	-2.95
6.29240	-7.43	-3.23	-7.43	-3.23	-7.43	-3.23	-7.43	-3.23
6.79422	-7.43	-3.53	-7.43	-3.53	-7.43	-3.53	-7.43	-3.53
7.38439	-7.43	-3.85	-7.43	-3.85	-7.43	-3.85	-7.43	-3.85
8.08642	-7.43	-4.20	-7.43	-4.20	-7.43	-4.20	-7.43	-4.20
8.93345	-7.43	-4.57	-7.43	-4.57	-7.43	-4.57	-7.43	-4.57
9.97410	-7.43	-4.96	-7.43	-4.96	-7.43	-4.96	-7.43	-4.96

FINAL INLET FLOWRATE: 33.0
FINAL LEAKAGE FLOWRATE: 40.4

FLUID SUBMODEL NAME = PIPE1 ; FLUID NO. = 9718

LOOPCT = 2
MAX TIME STEP = 1.000000E+15 ; LIMITING MESSAGE = NO LIMIT: TIME INDEPENDENT
LAST PSEUDO TIME STEP = 3600.00 VS. RSMAXF = 3600.00
CONVERGENCE STATUS = SUBMODEL CONVERGED AS OF 2 ITERATIONS

LUMP PARAMETER TABULATION FOR SUBMODEL PIPE1

LUMP	TYPE	TEMP	PRESSURE	QUALITY	DENSITY	ENTHALPY	HEAT RATE	MASS RATE	ENERGY RATE
98	JUNC	20.00	-238.8	0.	998.3	1.2962E+06	0.	1.9073E-06	2.000
99	JUNC	20.02	409.3	0.	998.3	1.2963E+06	0.	-1.9073E-06	-4.000
1	JUNC	20.00	8.7925E+04	0.	998.3	1.2963E+06	0.	0.	0.
2	JUNC	20.00	1.7609E+05	0.	998.3	1.2964E+06	0.	0.	0.
3	JUNC	20.00	2.6425E+05	0.	998.3	1.2964E+06	0.	0.	0.
4	JUNC	20.00	3.5242E+05	0.	998.3	1.2965E+06	0.	0.	0.
5	JUNC	20.00	4.4058E+05	0.	998.3	1.2966E+06	0.	0.	0.
6	JUNC	20.00	5.2874E+05	0.	998.3	1.2967E+06	0.	0.	0.
7	JUNC	20.00	6.1691E+05	0.	998.3	1.2968E+06	0.	0.	0.
8	JUNC	20.01	7.0507E+05	0.	998.3	1.2969E+06	0.	0.	0.
9	JUNC	20.01	7.9323E+05	0.	998.3	1.2970E+06	0.	0.	0.
10	JUNC	20.01	8.8140E+05	0.	998.3	1.2971E+06	0.	0.	0.
11	JUNC	20.01	8.7978E+05	0.	998.3	1.2971E+06	0.	0.	0.
12	JUNC	20.01	8.7654E+05	0.	998.3	1.2971E+06	0.	0.	0.
13	JUNC	20.01	8.7329E+05	0.	998.3	1.2971E+06	0.	0.	0.
14	JUNC	20.01	8.7005E+05	0.	998.3	1.2971E+06	0.	0.	0.
15	JUNC	20.01	8.6681E+05	0.	998.3	1.2971E+06	0.	0.	0.
16	JUNC	20.01	8.6357E+05	0.	998.3	1.2971E+06	0.	0.	0.
17	JUNC	20.01	8.6033E+05	0.	998.3	1.2971E+06	0.	0.	0.
18	JUNC	20.01	8.5709E+05	0.	998.3	1.2971E+06	0.	0.	0.
19	JUNC	20.01	8.5385E+05	0.	998.3	1.2971E+06	0.	0.	0.
20	JUNC	20.01	8.5223E+05	0.	998.3	1.2971E+06	0.	0.	0.
21	JUNC	20.02	7.6705E+05	0.	998.3	1.2970E+06	0.	0.	0.
22	JUNC	20.02	6.8187E+05	0.	998.3	1.2969E+06	0.	0.	0.
23	JUNC	20.02	5.9668E+05	0.	998.3	1.2968E+06	0.	0.	0.
24	JUNC	20.02	5.1150E+05	0.	998.3	1.2968E+06	0.	0.	0.
25	JUNC	20.02	4.2632E+05	0.	998.3	1.2967E+06	0.	0.	0.
26	JUNC	20.02	3.4114E+05	0.	998.3	1.2966E+06	0.	0.	0.
27	JUNC	20.02	2.5596E+05	0.	998.3	1.2965E+06	0.	0.	0.
28	JUNC	20.02	1.7077E+05	0.	998.3	1.2964E+06	0.	0.	0.
29	JUNC	20.02	8.5591E+04	0.	998.3	1.2963E+06	0.	0.	0.
1000	PLEN	20.00	0.	0.	998.3	1.2962E+06	0.	-22.45	-2.9095E+07
2000	PLEN	20.00	0.	0.	998.3	1.2962E+06	0.	22.45	2.9097E+07
3000	PLEN	20.00	0.	0.	998.3	1.2962E+06	0.	0.	0.

FLUID SUBMODEL NAME = PIPE1 ; FLUID NO. = 9718

LOOPCT = 2
MAX TIME STEP = 1.000000E+15 ; LIMITING MESSAGE = NO LIMIT: TIME INDEPENDENT
LAST PSEUDO TIME STEP = 3600.00 VS. RSMAXF = 3600.00
CONVERGENCE STATUS = SUBMODEL CONVERGED AS OF 2 ITERATIONS

PATH PARAMETER TABULATION FOR SUBMODEL PIPE1

PATH	TYPE	LMP 1	LMP 2	DUP I	DUP J	STAT	XL UPSTRM	FLOWRATE	DELTA PRES	FLOW REGIMES OR REYNOLDS NO.	
98	LOSS	1000	98	1.0	1.0	NORM	0.	22.45	238.8		
99	LOSS	99	2000	1.0	1.0	NORM	0.	22.45	409.3		
3000	CTLVLV	15	3000	1.0	1.0	NORM	0.	0.	8.6681E+05		
1	STUBE	98	1	1.0	1.0	NORM	0.	22.45	-8.8164E+04		1.49545E+05
2	STUBE	1	2	1.0	1.0	NORM	0.	22.45	-8.8164E+04		1.49548E+05
3	STUBE	2	3	1.0	1.0	NORM	0.	22.45	-8.8164E+04		1.49550E+05
4	STUBE	3	4	1.0	1.0	NORM	0.	22.45	-8.8164E+04		1.49553E+05
5	STUBE	4	5	1.0	1.0	NORM	0.	22.45	-8.8163E+04		1.49555E+05
6	STUBE	5	6	1.0	1.0	NORM	0.	22.45	-8.8163E+04		1.49558E+05
7	STUBE	6	7	1.0	1.0	NORM	0.	22.45	-8.8163E+04		1.49561E+05
8	STUBE	7	8	1.0	1.0	NORM	0.	22.45	-8.8163E+04		1.49563E+05
9	STUBE	8	9	1.0	1.0	NORM	0.	22.45	-8.8163E+04		1.49566E+05
10	STUBE	9	10	1.0	1.0	NORM	0.	22.45	-8.8163E+04		1.49571E+05
11	STUBE	10	11	1.0	1.0	NORM	0.	22.45	1620.	1.49571E+05	1.49571E+05
12	STUBE	11	12	1.0	1.0	NORM	0.	22.45	3241.	1.49571E+05	1.49574E+05
13	STUBE	12	13	1.0	1.0	NORM	0.	22.45	3241.	1.49574E+05	1.49579E+05
14	STUBE	13	14	1.0	1.0	NORM	0.	22.45	3241.	1.49579E+05	1.49582E+05
15	STUBE	14	15	1.0	1.0	NORM	0.	22.45	3241.	1.49582E+05	1.49584E+05
16	STUBE	15	16	1.0	1.0	NORM	0.	22.45	3241.	1.49584E+05	1.49587E+05
17	STUBE	16	17	1.0	1.0	NORM	0.	22.45	3241.	1.49587E+05	1.49592E+05
18	STUBE	17	18	1.0	1.0	NORM	0.	22.45	3241.	1.49592E+05	1.49595E+05
19	STUBE	18	19	1.0	1.0	NORM	0.	22.45	3241.	1.49595E+05	1.49597E+05
20	STUBE	19	20	1.0	1.0	NORM	0.	22.45	1620.	1.49597E+05	1.49600E+05
21	STUBE	20	21	1.0	1.0	NORM	0.	22.45	8.5182E+04	1.49600E+05	
22	STUBE	21	22	1.0	1.0	NORM	0.	22.45	8.5182E+04	1.49602E+05	
23	STUBE	22	23	1.0	1.0	NORM	0.	22.45	8.5182E+04	1.49602E+05	
24	STUBE	23	24	1.0	1.0	NORM	0.	22.45	8.5182E+04	1.49605E+05	
25	STUBE	24	25	1.0	1.0	NORM	0.	22.45	8.5182E+04	1.49610E+05	
26	STUBE	25	26	1.0	1.0	NORM	0.	22.45	8.5182E+04	1.49613E+05	
27	STUBE	26	27	1.0	1.0	NORM	0.	22.45	8.5182E+04	1.49616E+05	
28	STUBE	27	28	1.0	1.0	NORM	0.	22.45	8.5182E+04	1.49618E+05	
29	STUBE	28	29	1.0	1.0	NORM	0.	22.45	8.5182E+04	1.49621E+05	
30	STUBE	29	99	1.0	1.0	NORM	0.	22.45	8.5182E+04	1.49623E+05	

FLUID SUBMODEL NAME = PIPE6 ; FLUID NO. = 9716

MAX TIME STEP (GROWTH LIMITED) = 193.715 ; LAST CAUSE: TIE 8; NODE TEMPERATURE CHANGE LIMIT (WAS 48.43)
LAST TIME STEP = 30.0000 VS. DTMAXF/DTMINF = 1.000000E+30 / 0. ; AVERAGE TIME STEP = 29.3629
PROBLEM TIME TIMEN = 2730.00 VS. TIMEND = 2730.00

LUMP PARAMETER TABULATION FOR SUBMODEL PIPE6

LUMP	TYPE	TEMP	PRESSURE	QUALITY	DENSITY	ENTHALPY	HEAT RATE	MASS RATE	ENERGY RATE
1	TANK	1.001	8.8065E+04	0.	999.9	1.2165E+06	0.	0.	0.
2	TANK	1.002	1.7637E+05	0.	999.9	1.2166E+06	0.	0.	0.
3	TANK	1.002	2.6467E+05	0.	999.9	1.2166E+06	0.	0.	0.
4	TANK	1.003	3.5297E+05	0.	999.9	1.2167E+06	0.	0.	0.
5	TANK	1.004	4.4127E+05	0.	999.9	1.2168E+06	0.	0.	0.
6	TANK	1.004	5.2957E+05	0.	999.9	1.2169E+06	-8.181	0.	-4.181
7	TANK	1.005	6.1787E+05	0.	999.9	1.2170E+06	0.	0.	-4.000
8	TANK	1.006	7.0618E+05	0.	999.9	1.2171E+06	-8.181	0.	-10.18
9	TANK	1.007	7.9448E+05	0.	999.9	1.2172E+06	5.440	0.	-6.560
10	TANK	1.007	8.8278E+05	0.	999.9	1.2173E+06	10.55	0.	-11.45
11	TANK	1.009	8.8116E+05	0.	999.9	1.2173E+06	30.04	0.	-39.96
12	TANK	1.011	8.7791E+05	0.	999.9	1.2173E+06	43.96	0.	-104.0
13	TANK	1.015	8.7466E+05	0.	999.9	1.2173E+06	92.33	0.	-225.7
14	TANK	1.023	8.7141E+05	0.	999.9	1.2173E+06	206.7	0.	-439.3
15	TANK	1.036	8.6817E+05	0.	999.9	1.2174E+06	387.0	0.	-813.0
16	TANK	1.059	8.6492E+05	0.	999.9	1.2175E+06	668.8	0.	-1429.
17	TANK	1.097	8.6167E+05	0.	999.9	1.2176E+06	1080.	0.	-2386.
18	TANK	1.155	8.5843E+05	0.	999.9	1.2179E+06	1685.	0.	-3773.
19	TANK	1.243	8.5518E+05	0.	999.9	1.2183E+06	2516.	0.	-5696.
20	TANK	1.352	8.5356E+05	0.	999.9	1.2187E+06	3127.	0.	-7141.
21	TANK	1.500	7.6824E+05	0.	999.9	1.2192E+06	4193.	-1.9073E-06	-9659.
22	TANK	1.693	6.8292E+05	0.	999.9	1.2200E+06	5465.	1.9073E-06	-1.2697E+04
23	TANK	1.940	5.9760E+05	0.	999.9	1.2209E+06	6913.	0.	-1.6297E+04
24	TANK	2.248	5.1229E+05	0.	999.9	1.2221E+06	8560.	0.	-2.0390E+04
25	TANK	2.623	4.2697E+05	0.	999.9	1.2236E+06	1.0337E+04	0.	-2.4919E+04
26	TANK	3.069	3.4166E+05	0.	999.9	1.2254E+06	1.2223E+04	0.	-2.9783E+04
27	TANK	3.590	2.5634E+05	0.	999.9	1.2275E+06	1.4140E+04	0.	-3.4864E+04
28	TANK	4.185	1.7103E+05	0.	999.9	1.2300E+06	1.6059E+04	0.	-3.9955E+04
29	TANK	4.852	8.5716E+04	0.	999.9	1.2327E+06	1.7889E+04	0.	-4.4905E+04
98	JUNC	1.000	-236.9	0.	999.9	1.2164E+06	0.	0.	0.
99	JUNC	4.853	406.2	0.	999.9	1.2326E+06	0.	0.	0.
1000	PLEN	1.000	0.	0.	999.9	1.2164E+06	0.	-22.38	-2.7220E+07
2000	PLEN	20.00	0.	0.	998.3	1.2962E+06	0.	22.38	2.7584E+07
3000	PLEN	20.00	0.	0.	998.3	1.2962E+06	0.	0.	0.

FLUID SUBMODEL NAME = PIPE6 ; FLUID NO. = 9716

MAX TIME STEP (GROWTH LIMITED) = 193.715 ; LAST CAUSE: TIE 8; NODE TEMPERATURE CHANGE LIMIT (WAS 48.43)
LAST TIME STEP = 30.0000 VS. DTMAXF/DTMINF = 1.000000E+30 / 0. ; AVERAGE TIME STEP = 29.3629
PROBLEM TIME TIMEN = 2730.00 VS. TIMEND = 2730.00

PATH PARAMETER TABULATION FOR SUBMODEL PIPE6

PATH	TYPE	LMP 1	LMP 2	DUP I	DUP J	STAT	XL UPSTRM	FLOWRATE	DELTA PRES	FLOW REGIMES OR REYNOLDS NO.	
1	TUBE	98	1	1.0	1.0	NORM	0.	22.38	-8.8302E+04		85917.
2	TUBE	1	2	1.0	1.0	NORM	0.	22.38	-8.8302E+04		85920.
3	TUBE	2	3	1.0	1.0	NORM	0.	22.38	-8.8302E+04		85921.
4	TUBE	3	4	1.0	1.0	NORM	0.	22.38	-8.8302E+04		85923.
5	TUBE	4	5	1.0	1.0	NORM	0.	22.38	-8.8302E+04		85925.
6	TUBE	5	6	1.0	1.0	NORM	0.	22.38	-8.8302E+04		85926.
7	TUBE	6	7	1.0	1.0	NORM	0.	22.38	-8.8302E+04		85929.
8	TUBE	7	8	1.0	1.0	NORM	0.	22.38	-8.8302E+04		85930.
9	TUBE	8	9	1.0	1.0	NORM	0.	22.38	-8.8302E+04		85933.
10	TUBE	9	10	1.0	1.0	NORM	0.	22.38	-8.8302E+04		85935.
11	TUBE	10	11	1.0	1.0	NORM	0.	22.38	1624.	85935.	85939.
12	TUBE	11	12	1.0	1.0	NORM	0.	22.38	3247.	85939.	85945.
13	TUBE	12	13	1.0	1.0	NORM	0.	22.38	3247.	85945.	85956.
14	TUBE	13	14	1.0	1.0	NORM	0.	22.38	3247.	85956.	85978.
15	TUBE	14	15	1.0	1.0	NORM	0.	22.38	3247.	85978.	86014.
16	TUBE	15	16	1.0	1.0	NORM	0.	22.38	3247.	86014.	86077.
17	TUBE	16	17	1.0	1.0	NORM	0.	22.38	3247.	86077.	86180.
18	TUBE	17	18	1.0	1.0	NORM	0.	22.38	3247.	86180.	86342.
19	TUBE	18	19	1.0	1.0	NORM	0.	22.38	3247.	86342.	86586.
20	TUBE	19	20	1.0	1.0	NORM	0.	22.38	1623.	86586.	86890.
21	TUBE	20	21	1.0	1.0	NORM	0.	22.38	8.5318E+04	86890.	
22	TUBE	21	22	1.0	1.0	NORM	0.	22.38	8.5318E+04	87304.	
23	TUBE	22	23	1.0	1.0	NORM	0.	22.38	8.5317E+04	87854.	
24	TUBE	23	24	1.0	1.0	NORM	0.	22.38	8.5317E+04	88566.	
25	TUBE	24	25	1.0	1.0	NORM	0.	22.38	8.5316E+04	89469.	
26	TUBE	25	26	1.0	1.0	NORM	0.	22.38	8.5315E+04	90594.	
27	TUBE	26	27	1.0	1.0	NORM	0.	22.38	8.5314E+04	91973.	
28	TUBE	27	28	1.0	1.0	NORM	0.	22.38	8.5313E+04	93633.	
29	TUBE	28	29	1.0	1.0	NORM	0.	22.38	8.5312E+04	95606.	
30	TUBE	29	99	1.0	1.0	NORM	0.	22.38	8.5310E+04	97734.	
98	LOSS	1000	98	1.0	1.0	NORM	0.	22.38	236.9		
99	LOSS	99	2000	1.0	1.0	NORM	0.	22.38	406.2		
3000	CTLVLV	15	3000	1.0	1.0	NORM	0.	0.	8.6817E+05		

FLUID SUBMODEL NAME = PIPE6 ; FLUID NO. = 9718

MAX TIME STEP (GROWTH LIMITED) = 193.715 ; LAST CAUSE: TIE 8; NODE TEMPERATURE CHANGE LIMIT (WAS 48.43)
LAST TIME STEP = 30.0000 VS. DTMAXF/DTMINF = 1.000000E+30 / 0. ; AVERAGE TIME STEP = 29.3629
PROBLEM TIME TIMEN = 2730.00 VS. TIMEND = 2730.00

TIE PARAMETER TABULATION FOR SUBMODEL PIPE6

TIE	TYPE	UA	QTIE	LUMP	TEMP.	NODE	TEMP.	PATH 1	FRACT	PATH 2	FRACT
1	HTN	3.8050E+04	0.	1	1.001	WALL6.1	1.001	1	1.0000		
2	HTN	3.8051E+04	0.	2	1.002	WALL6.2	1.002	2	1.0000		
3	HTN	3.8051E+04	0.	3	1.002	WALL6.3	1.002	3	1.0000		
4	HTN	3.8052E+04	0.	4	1.003	WALL6.4	1.003	4	1.0000		
5	HTN	3.8052E+04	0.	5	1.004	WALL6.5	1.004	5	1.0000		
6	HTN	3.3508E+04	-8.181	6	1.004	WALL6.6	1.004	6	1.0000		
7	HTN	3.8053E+04	0.	7	1.005	WALL6.7	1.005	7	1.0000		
8	HTN	3.3509E+04	-8.181	8	1.006	WALL6.8	1.006	8	1.0000		
9	HTN	4.3214E+04	5.440	9	1.007	WALL6.9	1.007	9	1.0000		
10	HTN	4.3215E+04	10.55	10	1.007	WALL6.10	1.008	10	1.0000		
11	HTNC	4.8017E+04	30.04	11	1.009	WALL6.11	1.009	11	1.0000	12	0.5000
12	HTNC	4.8020E+04	43.96	12	1.011	WALL6.12	1.012	12	0.5000	13	0.5000
13	HTNC	4.8023E+04	92.33	13	1.015	WALL6.13	1.017	13	0.5000	14	0.5000
14	HTNC	4.8029E+04	206.7	14	1.023	WALL6.14	1.026	14	0.5000	15	0.5000
15	HTNC	4.8040E+04	387.0	15	1.036	WALL6.15	1.043	15	0.5000	16	0.5000
16	HTNC	4.8058E+04	668.8	16	1.059	WALL6.16	1.071	16	0.5000	17	0.5000
17	HTNC	4.8088E+04	1080.	17	1.097	WALL6.17	1.116	17	0.5000	18	0.5000
18	HTNC	4.8135E+04	1685.	18	1.155	WALL6.18	1.186	18	0.5000	19	0.5000
19	HTNC	4.8204E+04	2518.	19	1.243	WALL6.19	1.290	19	0.5000	20	1.0000
20	HTN	4.3461E+04	3127.	20	1.352	WALL6.20	1.417	21	1.0000		
21	HTN	4.3565E+04	4193.	21	1.500	WALL6.21	1.587	22	1.0000		
22	HTN	4.3701E+04	5465.	22	1.693	WALL6.22	1.806	23	1.0000		
23	HTN	4.3876E+04	6913.	23	1.940	WALL6.23	2.084	24	1.0000		
24	HTN	4.4096E+04	8560.	24	2.248	WALL6.24	2.427	25	1.0000		
25	HTN	4.4365E+04	1.0337E+04	25	2.623	WALL6.25	2.839	26	1.0000		
26	HTN	4.4690E+04	1.2223E+04	26	3.069	WALL6.26	3.325	27	1.0000		
27	HTN	4.5076E+04	1.4140E+04	27	3.590	WALL6.27	3.886	28	1.0000		
28	HTN	4.5525E+04	1.6059E+04	28	4.185	WALL6.28	4.520	29	1.0000		
29	HTN	4.5961E+04	1.7889E+04	29	4.852	WALL6.29	5.225	30	1.0000		

SUBMODEL NAME = WALL6

		CALCULATED			ALLOWED	
MAX DIFF DELTA T PER ITER	DRLXCC(	0)=	0.	VS. DRLXCA=	1.000000E-02	
MAX ARITH DELTA T PER ITER	ARLXCC(	0)=	0.	VS. ARLXCA=	1.000000E-02	
MAX DIFF DEL T PER TIME STEP	DTMPCC(WALL6	29)=	-0.310913	VS. DTMPCA=	2.00000	
MAX ARITH DEL T PER TIME STEP	ATMPCC(	0)=	0.	VS. ATMPCA=	1.000000E+30	
MIN STABILITY CRITERIA	CSGMIN(WALL6	29)=	37.5570			
MAX STABILITY CRITERIA	CSGMAX(WALL6	1)=	6.630527E+07			
NUMBER OF ITERATIONS	LOOPCT	=	1	VS. NLOOPT=	100	
PROBLEM TIME	TIMEN	=	2730.00	VS. TIMEND=	2730.00	
TIME STEP USED	DTIMEU	=	30.0000	VS. DTIMEI=	0.	

DIFFUSION NODES IN INPUT NODE NUMBER ORDER

T	1=	1.0008	T	2=	1.0017	T	3=	1.0020	T	4=	1.0030	T	5=	1.0036	T	6=	1.0044
T	7=	1.0051	T	8=	1.0059	T	9=	1.0066	T	10=	1.0075	T	20=	1.4169	T	21=	1.5866
T	22=	1.8064	T	23=	2.0844	T	24=	2.4267	T	25=	2.8392	T	26=	3.3253	T	27=	3.8860
T	28=	4.5203	T	29=	5.2249	T	11=	1.0087	T	12=	1.0119	T	13=	1.0171	T	14=	1.0264
T	15=	1.0431	T	16=	1.0711	T	17=	1.1165	T	18=	1.1863	T	19=	1.2897			

ARITHMETIC NODES IN INPUT NODE NUMBER ORDER
↔NONE↔

HEATER NODES IN INPUT NODE NUMBER ORDER
↔NONE↔

BOUNDARY NODES IN INPUT NODE NUMBER ORDER
↔NONE↔

MODEL = SINDA85 FLUINT SAMPLE PROBLEM 2 - SIPHON IN A ROMAN AQUADUCT

FLUID SUBMODEL NAME = PIPE1 ; FLUID NO. = 9718

MAX TIME STEP = 1.000000E+15 ; LIMITING MESSAGE = NO LIMIT: TIME INDEPENDENT
LAST TIME STEP = 1.04065 VS. DTMAXF/DTMINF = 1.000000E+30 / 0. ; AVERAGE TIME STEP = 0.434953
PROBLEM TIME TIMEN = 9.97410 VS. TIMEND = 9.97410

LUMP PARAMETER TABULATION FOR SUBMODEL PIPE1

LUMP	TYPE	TEMP	PRESSURE	QUALITY	DENSITY	ENTHALPY	HEAT RATE	MASS RATE	ENERGY RATE
98	JUNC	20.00	-515.2	0.	998.3	1.2962E+06	0.	0.	0.
99	JUNC	20.00	-44.83	0.	998.3	1.2962E+06	0.	0.	0.
1	JUNC	20.00	8.4299E+04	0.	998.3	1.2963E+06	0.	0.	0.
2	JUNC	20.00	1.6911E+05	0.	998.3	1.2964E+06	0.	0.	0.
3	JUNC	20.00	2.5393E+05	0.	998.3	1.2964E+06	0.	0.	0.
4	JUNC	20.01	3.3874E+05	0.	998.3	1.2965E+06	0.	0.	0.
5	JUNC	20.01	4.2356E+05	0.	998.3	1.2966E+06	0.	0.	0.
6	JUNC	20.01	5.0837E+05	0.	998.3	1.2967E+06	0.	0.	0.
7	JUNC	20.01	5.9319E+05	0.	998.3	1.2968E+06	0.	0.	0.
8	JUNC	20.01	6.7800E+05	0.	998.3	1.2969E+06	0.	0.	0.
9	JUNC	20.01	7.6282E+05	0.	998.3	1.2970E+06	0.	0.	0.
10	JUNC	20.02	8.4763E+05	0.	998.3	1.2971E+06	0.	0.	0.
11	JUNC	20.02	8.4415E+05	0.	998.3	1.2971E+06	0.	0.	0.
12	JUNC	20.02	8.3719E+05	0.	998.3	1.2971E+06	0.	0.	0.
13	JUNC	20.02	8.3023E+05	0.	998.3	1.2971E+06	0.	0.	0.
14	JUNC	20.02	8.2326E+05	0.	998.3	1.2971E+06	0.	0.	0.
15	JUNC	20.02	8.1755E+05	0.	998.3	1.2971E+06	0.	-1.9073E-06	-4.000
16	JUNC	20.00	8.1806E+05	0.	998.3	1.2970E+06	0.	-4.7684E-07	-1.000
17	JUNC	20.00	8.1843E+05	0.	998.3	1.2970E+06	0.	-4.7684E-07	-1.000
18	JUNC	20.00	8.1879E+05	0.	998.3	1.2970E+06	0.	-4.7684E-07	0.
19	JUNC	20.00	8.1916E+05	0.	998.3	1.2970E+06	0.	-4.7684E-07	-1.000
20	JUNC	20.00	8.1934E+05	0.	998.3	1.2970E+06	0.	0.	0.
21	JUNC	20.00	7.3740E+05	0.	998.3	1.2969E+06	0.	0.	0.
22	JUNC	20.00	6.5546E+05	0.	998.3	1.2968E+06	0.	0.	0.
23	JUNC	20.00	5.7352E+05	0.	998.3	1.2968E+06	0.	0.	0.
24	JUNC	20.00	4.9159E+05	0.	998.3	1.2967E+06	0.	0.	0.
25	JUNC	20.00	4.0965E+05	0.	998.3	1.2966E+06	0.	0.	0.
26	JUNC	20.00	3.2771E+05	0.	998.3	1.2965E+06	0.	0.	0.
27	JUNC	20.00	2.4577E+05	0.	998.3	1.2964E+06	0.	0.	0.
28	JUNC	20.00	1.6383E+05	0.	998.3	1.2963E+06	0.	0.	0.
29	JUNC	20.00	8.1893E+04	0.	998.3	1.2963E+06	0.	0.	0.
1000	PLEN	20.00	0.	0.	998.3	1.2962E+06	0.	-32.97	-4.2740E+07
2000	PLEN	20.00	0.	0.	998.3	1.2962E+06	0.	-7.428	-9.6285E+06
3000	PLEN	20.00	0.	0.	998.3	1.2962E+06	0.	40.40	5.2405E+07

MODEL = SINDA85 FLUINT SAMPLE PROBLEM 2 - SIPHON IN A ROMAN AQUADUCT

FLUID SUBMODEL NAME = PIPE6 ; FLUID NO. = 9718

MAX TIME STEP = 1.04065 ; LIMITING TUBE = 16 REASON = FLOWRATE CHANGE LIMIT
LAST TIME STEP = 1.04065 VS. DTMAXF/DTMINF = 1.000000E+30 / 0. ; AVERAGE TIME STEP = 0.434953
PROBLEM TIME TIMEN = 9.97410 VS. TIMEND = 9.97410

LUMP PARAMETER TABULATION FOR SUBMODEL PIPE6

LUMP	TYPE	TEMP	PRESSURE	QUALITY	DENSITY	ENTHALPY	HEAT RATE	MASS RATE	ENERGY RATE
1	TANK	20.00	8.3836E+04	0.	998.3	1.2963E+06	0.	-7.6294E-06	104.0
2	TANK	20.00	1.6826E+05	0.	998.3	1.2963E+06	0.	-7.6294E-06	116.0
3	TANK	20.00	2.5269E+05	0.	998.3	1.2964E+06	0.	-7.6294E-06	124.0
4	TANK	20.00	3.3712E+05	0.	998.3	1.2965E+06	0.	-3.8147E-06	132.0
5	TANK	20.00	4.2154E+05	0.	998.3	1.2966E+06	0.	-1.1444E-05	132.0
6	TANK	20.00	5.0597E+05	0.	998.3	1.2967E+06	0.	-3.8147E-06	104.0
7	TANK	20.00	5.9040E+05	0.	998.3	1.2968E+06	0.	-7.6294E-06	152.0
8	TANK	20.01	6.7482E+05	0.	998.3	1.2969E+06	-38.56	-3.8147E-06	65.44
9	TANK	20.01	7.5925E+05	0.	998.3	1.2970E+06	0.	-7.6294E-06	112.0
10	TANK	20.01	8.4368E+05	0.	998.3	1.2970E+06	0.	-7.6294E-06	128.0
11	TANK	20.01	8.3998E+05	0.	998.3	1.2970E+06	0.	-7.6294E-06	88.00
12	TANK	20.01	8.3259E+05	0.	998.3	1.2970E+06	-42.85	0.	85.15
13	TANK	20.01	8.2519E+05	0.	998.3	1.2970E+06	0.	-7.6294E-06	140.0
14	TANK	20.01	8.1780E+05	0.	998.3	1.2970E+06	0.	-1.1444E-05	140.0
15	TANK	20.01	8.1179E+05	0.	998.3	1.2970E+06	0.	-3.8147E-06	120.0
16	TANK	20.01	8.1286E+05	0.	998.3	1.2970E+06	0.	-1.4305E-06	17.50
17	TANK	20.01	8.1365E+05	0.	998.3	1.2970E+06	11.10	-1.4305E-06	29.10
18	TANK	20.01	8.1443E+05	0.	998.3	1.2970E+06	0.	-9.5367E-07	19.00
19	TANK	20.01	8.1521E+05	0.	998.3	1.2970E+06	0.	-4.7684E-07	10.00
20	TANK	20.01	8.1560E+05	0.	998.3	1.2970E+06	0.	-9.5367E-07	16.00
21	TANK	20.02	7.3404E+05	0.	998.3	1.2970E+06	0.	-1.4305E-06	16.50
22	TANK	20.02	6.5248E+05	0.	998.3	1.2969E+06	0.	-9.5367E-07	16.50
23	TANK	20.02	5.7091E+05	0.	998.3	1.2968E+06	0.	-4.7684E-07	16.50
24	TANK	20.02	4.8935E+05	0.	998.3	1.2967E+06	0.	-1.4305E-06	17.00
25	TANK	20.02	4.0779E+05	0.	998.3	1.2967E+06	0.	-4.7684E-07	16.50
26	TANK	20.02	3.2623E+05	0.	998.3	1.2966E+06	0.	-9.5367E-07	17.00
27	TANK	20.02	2.4467E+05	0.	998.3	1.2965E+06	0.	-1.4305E-06	16.50
28	TANK	20.02	1.6310E+05	0.	998.3	1.2964E+06	0.	-4.7684E-07	8.500
29	TANK	20.02	8.1542E+04	0.	998.3	1.2963E+06	0.	2.1935E-05	-383.5
98	JUNC	20.00	-590.3	0.	998.3	1.2962E+06	0.	0.	0.
99	JUNC	20.00	-19.87	0.	998.3	1.2962E+06	0.	0.	0.
1000	PLEN	20.00	0.	0.	998.3	1.2962E+06	0.	-35.30	-4.5751E+07
2000	PLEN	20.00	0.	0.	998.3	1.2962E+06	0.	-4.963	-6.4331E+06
3000	PLEN	20.00	0.	0.	998.3	1.2962E+06	0.	40.26	5.2218E+07

FLUID SUBMODEL NAME = PIPE1 ; FLUID NO. = 9718

MAX TIME STEP = 1.000000E-15 ; LIMITING MESSAGE = NO LIMIT: TIME INDEPENDENT
LAST TIME STEP = 1.04065 VS. DTMAXF/DTMINF = 1.000000E+30 / 0. ; AVERAGE TIME STEP = 0.434953
PROBLEM TIME TIMEN = 9.97410 VS. TIMEND = 9.97410

PATH PARAMETER TABULATION FOR SUBMODEL PIPE1

PATH	TYPE	LMP 1	LMP 2	DUP I	DUP J	STAT	XL UPSTRM	FLOWRATE	DELTA PRES	FLOW REGIMES OR REYNOLDS NO.	
98	LOSS	1000	98	1.0	1.0	NORM	0.	32.97	515.2		
99	LOSS	99	2000	1.0	1.0	NORM	0.	-7.428	-44.83		
3000	CTLVLV	15	3000	1.0	1.0	NORM	0.	40.40	8.1755E+05		
1	STUBE	98	1	1.0	1.0	NORM	0.	32.97	-8.4815E+04		2.19683E+05
2	STUBE	1	2	1.0	1.0	NORM	0.	32.97	-8.4815E+04		2.19691E+05
3	STUBE	2	3	1.0	1.0	NORM	0.	32.97	-8.4815E+04		2.19702E+05
4	STUBE	3	4	1.0	1.0	NORM	0.	32.97	-8.4815E+04		2.19710E+05
5	STUBE	4	5	1.0	1.0	NORM	0.	32.97	-8.4815E+04		2.19718E+05
6	STUBE	5	6	1.0	1.0	NORM	0.	32.97	-8.4814E+04		2.19729E+05
7	STUBE	6	7	1.0	1.0	NORM	0.	32.97	-8.4814E+04		2.19737E+05
8	STUBE	7	8	1.0	1.0	NORM	0.	32.97	-8.4814E+04		2.19748E+05
9	STUBE	8	9	1.0	1.0	NORM	0.	32.97	-8.4814E+04		2.19756E+05
10	STUBE	9	10	1.0	1.0	NORM	0.	32.97	-8.4814E+04		2.19764E+05
11	STUBE	10	11	1.0	1.0	NORM	0.	32.97	3481.	2.19764E+05	2.19768E+05
12	STUBE	11	12	1.0	1.0	NORM	0.	32.97	6962.	2.19768E+05	2.19779E+05
13	STUBE	12	13	1.0	1.0	NORM	0.	32.97	6962.	2.19779E+05	2.19787E+05
14	STUBE	13	14	1.0	1.0	NORM	0.	32.97	6962.	2.19787E+05	2.19798E+05
15	STUBE	14	15	1.0	1.0	NORM	0.	32.97	5710.	2.19798E+05	2.19783E+05
16	STUBE	15	16	1.0	1.0	NORM	0.	-7.428	-510.2	49513.	49490.
17	STUBE	16	17	1.0	1.0	NORM	0.	-7.428	-364.0	49490.	49490.
18	STUBE	17	18	1.0	1.0	NORM	0.	-7.428	-364.0	49490.	49490.
19	STUBE	18	19	1.0	1.0	NORM	0.	-7.428	-364.0	49490.	49490.
20	STUBE	19	20	1.0	1.0	NORM	0.	-7.428	-182.0	49490.	49490.
21	STUBE	20	21	1.0	1.0	NORM	0.	-7.428	8.1938E+04		49490.
22	STUBE	21	22	1.0	1.0	NORM	0.	-7.428	8.1938E+04		49490.
23	STUBE	22	23	1.0	1.0	NORM	0.	-7.428	8.1938E+04		49490.
24	STUBE	23	24	1.0	1.0	NORM	0.	-7.428	8.1938E+04		49490.
25	STUBE	24	25	1.0	1.0	NORM	0.	-7.428	8.1938E+04		49489.
26	STUBE	25	26	1.0	1.0	NORM	0.	-7.428	8.1938E+04		49489.
27	STUBE	26	27	1.0	1.0	NORM	0.	-7.428	8.1938E+04		49489.
28	STUBE	27	28	1.0	1.0	NORM	0.	-7.428	8.1938E+04		49489.
29	STUBE	28	29	1.0	1.0	NORM	0.	-7.428	8.1938E+04		49489.
30	STUBE	29	99	1.0	1.0	NORM	0.	-7.428	8.1938E+04		49489.

FLUID SUBMODEL NAME = PIPE6 ; FLUID NO. = 9718

MAX TIME STEP = 1.04065 ; LIMITING TUBE = 16 REASON = FLOWRATE CHANGE LIMIT
LAST TIME STEP = 1.04065 VS. DTMAXF/DTMINF = 1.000000E+30 / 0. ; AVERAGE TIME STEP = 0.434953
PROBLEM TIME TIMEN = 9.97410 VS. TIMEND = 9.97410

PATH PARAMETER TABULATION FOR SUBMODEL PIPE6

PATH	TYPE	LMP 1	LMP 2	DUP I	DUP J	STAT	XL UPSTRM	FLOWRATE	DELTA PRES	FLOW REGIMES OR REYNOLDS NO.	
1	TUBE	98	1	1.0	1.0	NORM	0.	35.30	-8.4427E+04		2.35153E+05
2	TUBE	1	2	1.0	1.0	NORM	0.	35.30	-8.4427E+04		2.35157E+05
3	TUBE	2	3	1.0	1.0	NORM	0.	35.30	-8.4427E+04		2.35161E+05
4	TUBE	3	4	1.0	1.0	NORM	0.	35.30	-8.4427E+04		2.35166E+05
5	TUBE	4	5	1.0	1.0	NORM	0.	35.30	-8.4427E+04		2.35170E+05
6	TUBE	5	6	1.0	1.0	NORM	0.	35.30	-8.4427E+04		2.35174E+05
7	TUBE	6	7	1.0	1.0	NORM	0.	35.30	-8.4427E+04		2.35178E+05
8	TUBE	7	8	1.0	1.0	NORM	0.	35.30	-8.4427E+04		2.35186E+05
9	TUBE	8	9	1.0	1.0	NORM	0.	35.30	-8.4427E+04		2.35191E+05
10	TUBE	9	10	1.0	1.0	NORM	0.	35.30	-8.4427E+04		2.35195E+05
11	TUBE	10	11	1.0	1.0	NORM	0.	35.30	3696.	2.35195E+05	2.35195E+05
12	TUBE	11	12	1.0	1.0	NORM	0.	35.30	7393.	2.35195E+05	2.35203E+05
13	TUBE	12	13	1.0	1.0	NORM	0.	35.30	7393.	2.35203E+05	2.35207E+05
14	TUBE	13	14	1.0	1.0	NORM	0.	35.30	7393.	2.35207E+05	2.35211E+05
15	TUBE	14	15	1.0	1.0	NORM	0.	35.30	6008.	2.35211E+05	2.35215E+05
16	TUBE	15	16	1.0	1.0	NORM	0.	-4.963	-1070.	33074.	33075.
17	TUBE	16	17	1.0	1.0	NORM	0.	-4.963	-781.8	33075.	33075.
18	TUBE	17	18	1.0	1.0	NORM	0.	-4.963	-781.8	33075.	33076.
19	TUBE	18	19	1.0	1.0	NORM	0.	-4.963	-781.8	33076.	33077.
20	TUBE	19	20	1.0	1.0	NORM	0.	-4.963	-390.9	33077.	33078.
21	TUBE	20	21	1.0	1.0	NORM	0.	-4.963	8.1562E+04		33078.
22	TUBE	21	22	1.0	1.0	NORM	0.	-4.963	8.1562E+04		33078.
23	TUBE	22	23	1.0	1.0	NORM	0.	-4.963	8.1562E+04		33079.
24	TUBE	23	24	1.0	1.0	NORM	0.	-4.963	8.1562E+04		33079.
25	TUBE	24	25	1.0	1.0	NORM	0.	-4.963	8.1562E+04		33080.
26	TUBE	25	26	1.0	1.0	NORM	0.	-4.963	8.1562E+04		33081.
27	TUBE	26	27	1.0	1.0	NORM	0.	-4.963	8.1562E+04		33082.
28	TUBE	27	28	1.0	1.0	NORM	0.	-4.963	8.1562E+04		33082.
29	TUBE	28	29	1.0	1.0	NORM	0.	-4.963	8.1562E+04		33082.
30	TUBE	29	99	1.0	1.0	NORM	0.	-4.963	8.1562E+04		33065.
98	LOSS	1000	98	1.0	1.0	NORM	0.	35.30	590.3		
99	LOSS	99	2000	1.0	1.0	NORM	0.	-4.963	-19.87		
3000	CTLVLV	15	3000	1.0	1.0	NORM	0.	40.26	8.1179E+05		

PROBLEM C: PRESSURE COOKER

This problem simulates a pressure cooker (autoclave) being refilled and returning to operation. This problem was chosen to demonstrate the following FLUINT features:

1. the properties of two-phase tanks
2. phase suction options
3. user-supplied heat transfer coefficients
4. user logic for specialized output and customized solutions
5. user-supplied Fortran subroutines

C.1 Problem Description

A 5 lb_m steel pressure cooker is operating at 15 psig, corresponding to a saturated steam temperature of 250°F (see Figure C–1). The cooker sits on a stove that is adding 500 W of heat to the base, and loses heat to the ambient air due to natural convection. The pressure is regulated by a backpressure control valve at the top of the cooker (a weight sitting on top of a vent). When the pressure is greater than the weight, the valve opens, releasing steam. The cooker can be refilled by injecting pressurized ambient (70°F) liquid water. The effects of any pot roast or other nonfluid contents are ignored. The relevant characteristics are:

Cooker:
- Operating pressure ..... 15 psig
- Volume ..... 0.5 ft^3
- Base (boiling) area ..... 1 ft^2
- Surface (loss) area ..... 3.5 ft^2
- Initial liquid ..... 1/2 cup
- Wall mass ..... 5 lb_m steel
- Stove heat ..... 500 W

Injection source (ambient):
- Temperature ..... 70°F
- Pressure ..... 15 psig

Valve:
- Open flow area ..... 0.005 sq. in.
- Opening pressure ..... 15 psig.

The heat transfer coefficient for natural convection to the ambient are is estimated to be 1.1 BTU/hr-ft^2-°F.

At time zero, the injector begins to supply liquid to the cooker such that the total mass of the cooker doubles in 30 seconds. The sudden mixing with cool water causes the pressure in the cooker to quickly drop. The response of the cooker is desired, including how far the pressure drops, and how long it takes for the pressure to come back up to operating conditions. Also, the performance of the oscillatory pressure control valve is desired.

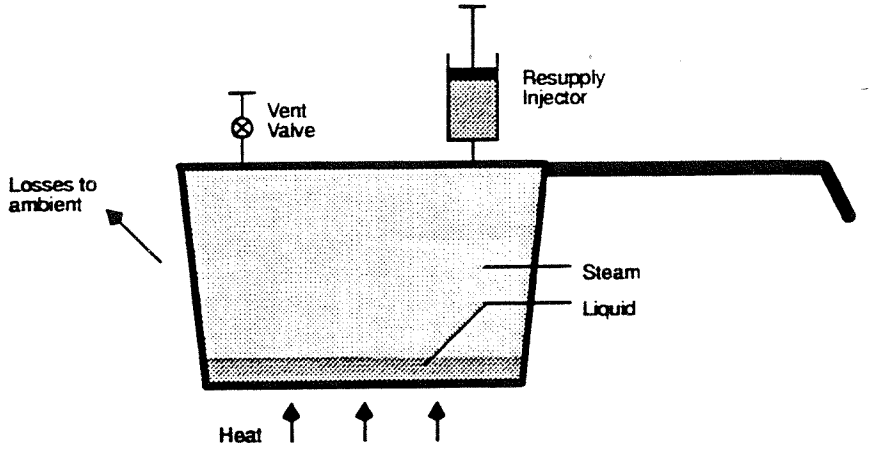


Figure C–1 Pressure Cooker and Resupply Injector

C.2 A FLUINT Model

A rigid FLUINT tank will be used to model the pressure cooker itself. The selection of a single tank to simulate the cooker carries the hidden assumption of perfect mixing within the cooker. Imperfect mixing can also be simulated via the NONEQ0 and NONEQ1 simulation subroutines, as demonstrated in Sample Problem F.

This tank will be tied to a diffusion node representing the wall. The use of a single node precludes the effects of wall gradients, although these could be included if sufficient information were available. The heat transfer phenomena will be pool boiling; a forced convection HTN or HTNC tie would be inappropriate, so an HTU tie will be used whose UA value is calculated using Rohsenow's pool boiling correlation:

$$UA = \left(\frac{C_p}{C_{sf}\, Pr^{1.7}} \right)^3 \left(\frac{T_{wall} - T_{sat}}{h_{fg}} \right)^2 \mu \sqrt{\frac{g(\varrho_l - \varrho_v)}{\sigma}}$$

where:

UA	heat transfer coef. times area
C_p	liquid specific heat
C_{sf}	surface roughness coefficient (about 0.0132)
Pr	liquid Prandtl number
T_{wall}	wall temperature
T_{sat}	fluid (saturation) temperature
h_{fg}	heat of vaporization
μ	liquid viscosity
g	acceleration of gravity
ϱ_l	liquid density
ϱ_v	vapor density
σ	surface tension

This correlation is not linear with temperature; the value must be continuously updated as a function of wall and fluid properties in FLOGIC 0.

The convective losses to ambient are represented with a 70°F boundary node and a linear conductor equal to the hA product connecting that node to the tank wall. Recall that the thermal model cannot operate in steady-state without a boundary node (unless DRPMOD is used), and that no nonboundary node can be isolated (zero sum of conductors). Therefore even if there were negligible losses to ambient, a boundary node would still be needed along with a arbitrarily small but nonzero conductor to the cooker wall.

The resupply injection occurs at a constant rate; an MFRSET connector will be used to transport the liquid from a plenum containing the supply liquid. As an alternate, a liquid tank with a negative VDOT (representing squeezing by a piston) could be used as the injector. This would be inappropriate unless the pressure response of the injector is required.

As usual, the choices for the valve are more difficult. First, for simplicity it is assumed that the valve is not choked (a call to the CHOKER routine could be used otherwise, or just in case conditions changed and choking ever did occur). A check valve model might be considered to open and close the valve automatically. However, there is currently no offset pressure available as an option in the CHKVLV model, so a CTLVLV will be used instead. Without knowing more about the valve, an open FK value of 2.0 is selected. Note that such an oscillating control valve would be unacceptable in an incompressible liquid system because of pressure surges. Because the valve is on top of the cooker, it will be directed to extract only vapor from the cooker using the phase suction options.

C.3 The Input File

The input file is listed immediately following the last section.

OPTIONS DATA—As with previous examples, a user file (USER1) is named that will contain a summary listing of the output file.

CONTROL DATA—English units are chosen, with units of °F and psig. The value of OUTPTF is given as 3 seconds, although this will be modified to reduce output during the recovery phase of the event.

USER DATA—Variables to be used in the logic are declared and initialized. Among them is a flag to be used by the logic to distinguish the last minute of the simulation, when the cooker is allowed to run with the control valve oscillating. Also present are variables used to contain statistics about the pressure history during the last minute.

NODE DATA—One thermal submodel named WALL is used. It contains a boundary node (#999) set at ambient temperature, and a diffusion node (#1) representing the steel wall.

CONDUCTOR DATA—One linear conductor (#1) is given between the ambient and the diffusion node, and is set equal to the natural convection heat transfer coefficient times the surface area.

SOURCE DATA—The heat source term of 500 W is placed on the wall node.

FLOW DATA-One fluid submodel named COOKER is used. The working fluid is "718", meaning the standard library description of water. The default pressure is set high such that it is overridden later. The default initial flowrate is zero.

The names and types of the elements are as follows:

TANK 1	pressure cooker
TIE 1	pool boiling tie from wall node to fluid
CTLVLV 1	relief valve
PLENUM 2	exhaust to atmosphere
MFRSET 3	injector
PLENUM 3	injection source

Note that air cannot be included in the model directly; the exhaust is assumed to be steam at room temperature and pressure.

FLOGIC 0—The pool boiling heat transfer UA is adjusted with a call to PBOIL, a routine supplied within SUBROUTINE DATA. Note that the last argument, the fluid identifier, will be translated into a location in integer array FI. As such, it can be passed to the subroutine for further property routine calls without further declarations of "fsmn.FI" or "FInnn." This trick means that the subroutine PBOIL can be used in any model, independent of the working fluid. This logic block also contains the logic necessary to open or close the vent valve.

FLOGIC 2—This logic block contains three calculations for the transient run. First, pressure statistics are calculated during the last minute of the simulation. This includes the maximum and minimum pressure, and the time-averaged pressure that is found by integrating the pressure profile and dividing by the total time. The control constant DTIMUF, the size of the time step just taken, is used for this calculation.

Second, the code checks to see when the cooker has recovered from the injection, signaling the last minute of the simulation and resetting the output interval to be 3 seconds. Third, preventative logic is added to stop the simulation if the cooker ever dries out.

OUTPUT CALLS—This logic block defines the output to be produced during the transient run at each output interval. This data includes temperatures, pressures, and flowrates. Note that a compressed line of important data is written to the user file USER1.

OPERATIONS DATA—The configuration is built, containing both submodels. Also, the default model name is set to COOKER to avoid extra input. The first order of business is to change the mass in the cooker to comply with the initial conditions of 1/2 cup (or about 1/4 lb_m). The tank state was input using TL and a guessed XL. Logic is added here to estimate the desired quality using a call to the VSV (vapor specific volume) routine for water. This quality is then imposed on the tank using the CHGLMP routine.

Second, an initial cooker wall temperature is calculated with a preliminary call to PBOIL, assuming constant convective losses. An alternate method would have been to call FASTIC or STDSTL and to initialize this single temperature. However, there are two difficulties with that approach. First, *no steady-state exists for the fluid submodel.* The routines HLDLMP and

RELLMP would have to be used to make the tank act like a plenum. More importantly, *the UA calculation is extremely sensitive to temperature differences*. The heat rate is even more sensitive. Because the UA values of HTU ties are held constant each solution step, wild oscillations in the wall temperature would result with no capacitance term (i.e., during steady-state solutions). The user would have to add logic to keep the iterations damped and within reasonable bounds. The best method for such added control would be to solve for the UA value and the resulting wall temperature implicitly in the FLOGIC 0 block; such calculations would be very similar to the single-pass calculation made in OPERATIONS DATA.

Third, the fill transient is run as one transient solution for 30 seconds. The second transient is then the recovery response, which will contain one minute of "steady" operation after recovery. During the recovery response, the output interval is increased tenfold to reduce the volume of output. During the "steady" last minute, the time step is limited to 1 second to keep the control valve logic updated (prevent pressure overshoots and undershoots due to lack of control).

SUBROUTINE DATA—The pool boiling calculation is contained within subroutine PBOIL, which is located within this block. Note the internal use of SI units, and the use of the last argument for property routine calls. The translation (and therefore the debug option) has been turned off for this entire segment with the FSTART command. If SINDA/FLUINT translatable variables (T, G, UA, TL, QDOT, etc.) had been required, a CALL COMMON command could have been placed after the "SUBROUTINE PBOIL" declaration.

C.4 Output Description

The user summary file and the selected portions of the OUTPUT file are listed following the input file. Figure C–2 contains a plot of the cooker pressure history.

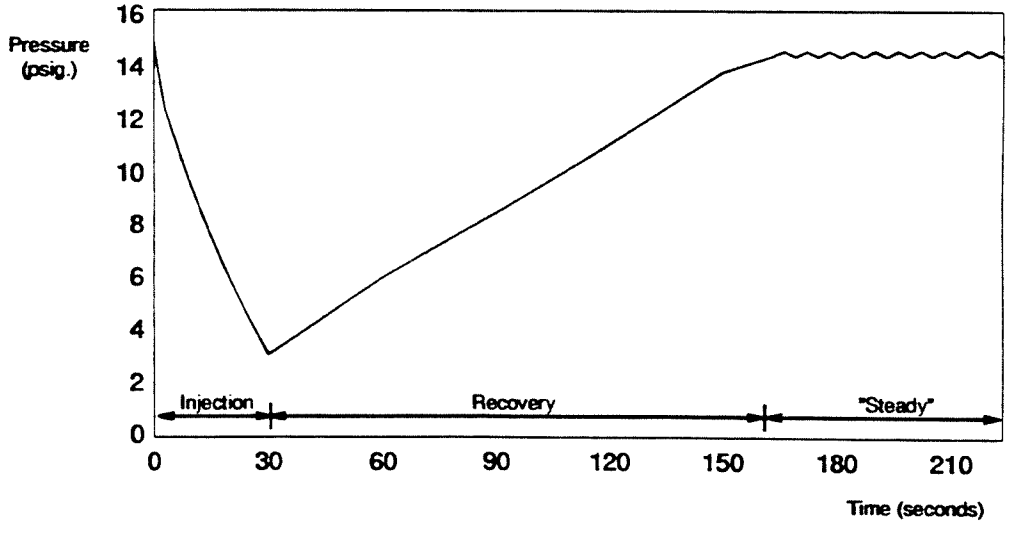


Figure C–2 Pressure History of the Cooker

The injection obviously has a dramatic effect on the cooker because so much mass is being added but so little energy. Unlike an open (constant pressure) pot, the constant volume cooker stays in a saturated state. The pressure drops to about 3 psig, and takes about 2 minutes to recover to the normal value. This occurs despite the fact that the heat into the fluid more than doubles due to the lag of the warm wall. Without the mass of the cooker included, the pressure would drop to about -2 psig.

This is an important trait to remember when modeling two-phase fluid systems: *because of the perfect mixing assumption, significant pressure fluctuations may be caused by irregular flowrates and heat rates.* In the real system, the cooler liquid would not be in equilibrium with the vapor phase until sufficient mass and energy had been exchanged over the course of a few seconds. By assuming perfect mixing, FLUINT assumes that this nonequilibrium stage is negligibly short. Thus, the cooker responds faster than it would in reality. This assumption is normally valid and it greatly expedites the numeric solution. Furthermore, the nonequilibrium exchange of mass and energy is often difficult to characterize with any accuracy. With a few assumptions regarding the physical processes involved, the user may wish to attempt such modeling using the NONEQ0 and NONEQ1 routines.

Some warnings are produced by the time step controller when changes are rapid: when the filling starts and when it stops. Strictly, a smaller time step is required at these two points in order to guarantee accuracy (DRLXCA or 0.01 degrees error per time step). However, because these two events are brief, relatively little error accumulates, and a reduction in DTIMEH (or DTMAXF) produces the same results. If such changes had been more common, either a greater error could be accepted or the time step could have been restricted.

With respect to the operation of the cooker at "steady-state," the peak pressure was 15.1 psig, the lowest pressure was 14.5 psig, and the time-averaged pressure was 14.8 psig. The flowrate of the valve when open was 4.3 lb_m/hr of steam. Comparing that to the total mass lost over the 60 second interval, it can be calculated that the valve was open an average of 24% percent of the time. Thus, the true pressure profile can be deduced to be a triangular wave with sudden pressure decreases and gradual pressure increases. The profile shown by the output plot is incorrect due to sampling errors: the output is sampled at a frequency smaller than the valve frequency, which is unknown. The user should always be aware of the fact that *perceived* behavior in a time history may be due to the output interval and not to any real model behavior.

In a real pressure cooker the valve area oscillates, but is it always open as long as the pressure is sufficient to lift the counterweight. Such a simulation could be done by calculating the loss factor and/or flow area within the logic blocks as a function of pressure according to a much more detailed dynamic model of the valve/weight system.

Finally, the sensitivity of the pool boiling correlation can be seen in the heat rate tabulated for the last minute of "steady" operation. The power into the fluid fluctuated quickly from 230W to 320W, largely due to fluctuations in the saturation temperature as the system pressure oscillated.

In many ways, this sample problem represents a simplification of the storage tank analyzed in Sample Problem F. In that analysis, wall gradients are includes, and nonequilibrium methods are used to analyze a liquid fill process. The reader should find that comparisons of the differences in modeling and in predicted behavior are very educational.

Input File

```
HEADER OPTIONS DATA
TITLE FLUINT SAMPLE PROBLEM 3 - PRESSURE COOKER
         MODEL    = PCOOK
C
         OUTPUT   = prescook.out
         USER1    = prescook.usr
C
HEADER CONTROL DATA,GLOBAL
         UID      = ENG
         ABSZRO   = -459.6
         PATMOS   = -14.7
         OUTPTF   = 3.0/3600.0                $ 3 SEC TO START
         OUTPUT   = 1000.0                    $ JUST TO SATISFY SINDA
C
HEADER USER DATA, GLOBAL
         SMAS     = 0.0                       $ MASS IN COOKER
         IFINAL   = 0                         $ SIGNAL FOR LAST MINUTE
         PMAX     = 0.0                       $ MAX PRESSURE
         PMIN     = 1000.0                    $ MIN PRESSURE
         POLD     = 15.0                      $ LAST PRESSURE
         PAVG     = 0.0                       $ TIME AVGD PRESSURE
C
HEADER NODE DATA, WALL
         1,260.0,5.0*0.11                    $ COOKER WALL
         -999,70.0,0.0                       $ AMBIENT
HEADER CONDUCTOR DATA, WALL
         1,1,999,1.1*3.5                     $ NATL CONV TO AIR
HEADER SOURCE DATA, WALL
         1,500.0/0.293                       $ 500 W HEAT INPUT
C
```

```
HEADER FLOW DATA,COOKER,FID=718                    $ USE LIBRARY WATER
C
LU DEF,            PL       = 1000.0               $ MAKE SURE PRESSURE
C                                                  $ OVERWRITTEN
PA DEF,            FR       = 0.0
C
LU TANK,1,         VOL      = 0.5                  $ COOKER
                   XL       = 0.1                  $ GUESS
                   TL       = 250.0
T HTU,1,1,WALL.1                                   $ POOL BOILING TIE
                   UA       = 100.0                $ GUESS
PA CONN,1,1,2,     STAT     = VS                   $ VAPOR SUCTION
                   DEV      = CTLVLV               $ RELEASE VALVE
                   FK       = 2.0                  $ GUESS
                   AF       = 0.005/144.0          $ GUESS
LU PLEN,2,         TL       = 212.0                $ STEAM EXHAUST
                   XL       = 1.0
LU PLEN,3,         TL       = 70.0                 $ INJECTION SOURCE
                   PL       = 15.0
                   XL       = 0.0
PA CONN,3,3,1,     DEV      = MFRSET               $ INJECTOR (SHUT)
C
HEADER OUTPUT CALLS, COOKER                        $ OUTPUT OPERATIONS
         CALL LMPTAB('ALL')
         CALL TIETAB('ALL')
         CALL PTHTAB('ALL')
C
C NOW, THE CONDENSED TABLE OF PARAMETERS ON USER FILE
C FLOWRATES USE SPECIAL FORMAT TO NEGLECT NEAR-ZERO VALUES
C
         SMAS              = VOL1*DL1
         WRITE(NUSER1,10) 3600.0*TIMEN,PL(1),XL(1),TL(1),WALL.T(1)
                  ,QDOT(1)*0.293,SMAS,FR(1),FR(3)
F10      FORMAT(1X,7(1PG12.4,2X),0P,2(F12.2,2X))
C
```

```
HEADER FLOGIC 0, COOKER                          $ LOGIC BLOCK
C
C UPDATE UA ACCORDING TO PBOIL ROUTINE
C
        CALL PBOIL(UTEST,PL1-PATMOS,TL1-ABSZRO,WALL.T1-ABSZRO,
     .             .TRUE.,COOKER.FI)
        UA1              = UTEST*1.0        $ FACTOR IN AREA
C
C       THIS LOGIC ADJUSTS THE VALVE AREA TO MAINTAIN
C       A "CONSTANT" BACK PRESSURE IN THE TANK, CLOSING
C       THE VALVE IF PRESSURE DROPS BELOW ATMOSPHERIC
C       THIS IS USED ONLY FOR THE LAST MINUTE
C
        IF(PL1 .LE. 15.0D0)THEN
                FK1              = -1.0
        ELSE
                FK1              = 2.0
        ENDIF
C
HEADER FLOGIC 2, COOKER                          $ LOGIC BLOCK
C
C DURING LAST MINUTE, SAVE UP MAX AND MINIMUM PRESSURES
C
        IF(IFINAL .EQ. 1)THEN
                PAVG     = PAVG + (SNGL(PL1) + POLD)*DTIMUF*0.5
                POLD     = SNGL(PL1)
                PMAX     = MAX(PMAX,POLD)
                PMIN     = MIN(PMIN,POLD)
        ENDIF
C
C       THIS LOGIC STOPS THE TRANSIENT IF DRIED OUT.
C       IF VALVE OPENS AGAIN (PRESSURE AT 15 PSIG), THEN
C       COMMENCE FINAL MINUTE OF SIMULATION
C
        IF(FK1 .GT. 0.0 .AND. TIMEN .GT. 30.0/3600.0 .AND.
     .     IFINAL .NE. 1)THEN
                TIMEND           = TIMEN + 60.0/3600.0
                IFINAL           = 1
                OUTPTF           = 3.0/3600.0
                WRITE(NUSER1,10) TIMEN*3600.0
10              FORMAT(/' COOKER AT FULL PRES. AT TIME = ',1pG13.4)
        ELSEIF(XL1 .GE. 1.0)THEN
                TIMEND           = TIMEN
                WRITE(NUSER1,20) TIMEN*3600.0
20              FORMAT(/' COOKER DRIED OUT AT TIME = ',1pG13.4)
        ENDIF
C
```

```
HEADER OPERATIONS DATA                          $ LIST OPERATIONS
C
BUILDF CONFIG,COOKER
BUILD CONFIG,WALL
C
DEFMOD COOKER
C
C FIRST, RESET COOKER QUALITY SUCH THAT THERE IS 1/2 CUP WATER
C OR ABOUT 1/4 LBM OF WATER (APPROX EQUAL TO TOTAL MASS)
C
         XTEST    = VOL1/(0.25*VSV(PL1-PATMOS,TL1-ABSZRO,FI718))
         CALL CHGLMP('COOKER',1,'XL',XTEST,'TL')
C
C FIND INITIAL WALL TEMPERATURE BASED ON EST. LEAK TO AMBIENT
C
         CALL PBOIL(UTEST,PL1-PATMOS,TL1-ABSZRO,WALL.T1-ABSZRO,
     .             .TRUE.,COOKER.FI)
         UA1                = UTEST*1.0       $ FACTOR IN AREA
         DTEST              = WALL.T1-TL1
         UTEST              = UA1/DTEST**2
DEFMOD WALL
         QTEST              = Q1 - G1*(T1-T999)
         DTEST              = (QTEST/UTEST)**(0.3333)
         T1                 = DTEST + COOKER.TL1
C
C THESE NEXT LINES SIMPLY INITIALIZE FOR OUTPUT PURPOSES
C
DEFMOD COOKER
         UA1                = UTEST*DTEST**2
         QDOT1              = UA1*(WALL.T1 - TL1)
C
C WRITE OUTPUT HEADER
C
         WRITE(NUSER1,10)
FSTART
10       FORMAT(/'   TIME (SEC)',T19,'P COOKER',6X,'X COOKER',6X
     .   ,'T COOKER',6X,'T WALL',8X,'HEAT (W)',6X,'MASS INSIDE',
     .            5X,'VENT RATE',5X,'FILL RATE'/)
FSTOP
C
C FIRST TRANSIENT:
C INJECT SO AS TO DOUBLE TOTAL MASS IN 30 SEC
C
         SMAS               = VOL1*DL1
         TIMEND             = 30.0/3600.0
         SMFR3              = SMAS/TIMEND
         CALL FWDBCK
C
```

```
C NOW SHUT OFF INJECTOR AND HEAT UP UNTIL VALVE OPENS OR ONE HOUR,
C OR DRYOUT
C
         SMFR3              = 0.0
         TIMEND             = 1.0
         OUTPTF             = 30.0/3600.0      $ INCREASE INTERVAL
         DTMAXF             = 1.0/3600.0       $ LIMIT STEP TO 1 SEC
C                                               $ TO CAPTURE P > 15
         CALL FWDBCK
C
C WRITE OUT SUMMARY OF LAST MINUTE
C
         WRITE(NUSER1,20)PMAX,PMIN,PAVG*3600.0/60.0
FSTART
20       FORMAT(/' THE MAXIMUM PRESSURE WAS:',T40,1PG13.4/
     .          ' THE MINIMUM PRESSURE WAS:',T40,G13.4/
     .          ' THE TIME-AVG PRESSURE WAS:',T40,G13.4)
FSTOP
C
```

```
HEADER SUBROUTINE DATA
C
FSTART
        SUBROUTINE PBOIL(U,P,T,TW,ENG,IDF)
C
C POOL BOILING CORRELATION FROM ROHSENHOW
C U     RETURNED HEAT TRANS COEF
C P     SAT PRESSURE
C T     SAT TEMP
C TW    WALL TEMP  (TW > T ASSUMED)
C ENG   UNIT FLAG: TRUE IF ENGLISH
C IDF   TRANSLATED PROPERTY ARRAY
C
        DIMENSION IDF(1)
        DOUBLE PRECISION P
        LOGICAL ENG
C
        DATA CSF/0.0132/,GEE/9.806/
        SAVE CSF,GEE
C
C CP LIQ
        CPL             = VCPF(P,T,IDF)
C DEN LIQ
        RHOL            = VDL(T,IDF)
C DEN VAP
        RHOV            = 1.0/VSV(P,T,IDF)
C HEAT OF VAPORIZ.
        HFG             = VHFG(P,T,1.0/RHOV,IDF)
C VISC LIQ
        VISL            = VVISCF(T,IDF)
C SURF TENS
        STEN            = VST(T,IDF)
C THERMAL COND
        TCONDL          = VCONDF(T,IDF)
C
C CONVERT FOR ENGLISH UNITS
C
        DELT            = TW - T
        IF(ENG)THEN
                CPL             = CPL*4187.0
                RHOL            = RHOL*16.018
                RHOV            = RHOV*16.018
                HFG             = HFG*2326.1
                VISL            = VISL*1.488/3600.0
                STEN            = STEN/0.3048*4.448
                TCONDL          = TCONDL*1.731
                DELT            = DELT/1.8
        ENDIF
```

```
C
C PRANDTL NUMBER TERM
         PRTERM            = CSF*(CPL*VISL/TCONDL)**(1.7)
C
         U                 = ( CPL/PRTERM )**3 *
     .                       ( DELT/HFG )**2 * VISL *
     .                       SQRT( GEE*(RHOL-RHOV)/STEN )
C
C CONVERT BACK TO ENGLISH IF NEEDED
C
         IF(ENG) U                 = U/5.678
C
         RETURN
         END
C
END OF DATA
```

Processor Output

TIME (SEC)	P COOKER	X COOKER	T COOKER	T WALL	HEAT (W)	MASS INSIDE	VENT RATE	FILL RATE
0.	15.04	0.1446	250.0	257.0	285.5	0.2482	0.00	0.00
3.000	12.63	0.1216	245.2	255.5	717.9	0.2723	0.00	29.78
6.000	11.28	0.1062	242.4	252.7	820.6	0.2971	0.00	29.78
9.000	10.03	9.3474E-02	239.6	250.1	810.2	0.3219	0.00	29.78
12.00	8.849	8.2836E-02	236.9	247.5	795.1	0.3468	0.00	29.78
15.00	7.755	7.3867E-02	234.3	245.0	784.3	0.3716	0.00	29.78
18.00	6.733	6.6227E-02	231.8	242.5	772.5	0.3964	0.00	29.78
21.00	5.777	5.9666E-02	229.4	240.2	758.9	0.4212	0.00	29.78
24.00	4.884	5.3994E-02	227.0	237.9	746.9	0.4460	0.00	29.78
27.00	4.051	4.9063E-02	224.7	235.7	735.3	0.4708	0.00	29.78
30.00	3.273	4.4756E-02	222.5	233.5	731.3	0.4957	0.00	29.78
30.00	3.273	4.4756E-02	222.5	233.5	731.3	0.4957	0.00	29.78
60.00	6.194	5.1536E-02	230.4	237.3	195.0	0.4957	0.00	0.00
90.00	8.701	5.7312E-02	236.6	243.2	193.1	0.4957	0.00	0.00
120.0	11.33	6.3319E-02	242.5	248.8	191.0	0.4957	0.00	0.00
150.0	14.06	6.9539E-02	248.1	254.3	189.1	0.4957	0.00	0.00
COOKER AT FULL PRES. AT TIME = 162.0								
162.0	14.59	7.0910E-02	249.1	256.4	188.4	0.4945	4.31	0.00
165.0	14.78	7.1436E-02	249.5	256.3	325.9	0.4939	0.00	0.00
168.0	14.54	7.1064E-02	249.0	256.6	221.9	0.4927	4.30	0.00
171.0	14.78	7.1679E-02	249.5	256.3	343.8	0.4922	0.00	0.00
174.0	14.54	7.1318E-02	249.0	256.6	223.8	0.4910	4.30	0.00
177.0	14.78	7.1943E-02	249.5	256.3	345.6	0.4905	0.00	0.00
180.0	14.55	7.1582E-02	249.0	256.6	224.1	0.4893	4.30	0.00
183.0	14.79	7.2212E-02	249.5	256.3	345.6	0.4887	0.00	0.00
186.0	14.55	7.1848E-02	249.0	256.6	224.1	0.4875	4.30	0.00
189.0	14.79	7.2481E-02	249.5	256.4	345.7	0.4870	0.00	0.00
192.0	14.55	7.2116E-02	249.1	256.6	223.9	0.4858	4.30	0.00
195.0	14.79	7.2753E-02	249.5	256.4	345.7	0.4853	0.00	0.00
198.0	14.56	7.2385E-02	249.1	256.6	223.9	0.4841	4.30	0.00
201.0	14.80	7.3026E-02	249.5	256.4	345.9	0.4836	0.00	0.00
204.0	14.56	7.2656E-02	249.1	256.6	224.0	0.4824	4.30	0.00
207.0	14.80	7.3302E-02	249.5	256.4	346.3	0.4818	0.00	0.00
210.0	14.56	7.2930E-02	249.1	256.6	223.6	0.4806	4.30	0.00
213.0	14.80	7.3578E-02	249.5	256.4	345.7	0.4801	0.00	0.00
216.0	14.57	7.3204E-02	249.1	256.6	223.6	0.4789	4.30	0.00
219.0	14.81	7.3856E-02	249.5	256.4	345.7	0.4784	0.00	0.00
222.0	14.57	7.3481E-02	249.1	256.6	223.6	0.4772	4.30	0.00
222.0	14.57	7.3481E-02	249.1	256.6	285.8	0.4772	1.93	0.00

THE MAXIMUM PRESSURE WAS: 15.06
THE MINIMUM PRESSURE WAS: 14.42
THE TIME-AVG PRESSURE WAS: 14.73

MODEL = PCOOK FLUINT SAMPLE PROBLEM 3 - PRESSURE COOKER
FWDBCK

FLUID SUBMODEL NAME = COOKER ; FLUID NO. = 718

MAX TIME STEP = 0. ; LIMITING MESSAGE = (NO SOLUTION STEP TAKEN YET)
LAST TIME STEP = 0. VS. DTMAXF/DTMINF = 1.000000E+30 / 0. ; AVERAGE TIME STEP = 0.
PROBLEM TIME TIMEN = 0. VS. TIMEND = 8.333334E-03

LUMP PARAMETER TABULATION FOR SUBMODEL COOKER

LUMP	TYPE	TEMP	PRESSURE	QUALITY	DENSITY	ENTHALPY	HEAT RATE	MASS RATE	ENERGY RATE
1	TANK	250.0	15.04	0.1446	0.4964	357.2	974.4	0.	974.4
2	PLEN	212.0	-4.0171E-02	1.000	3.7257E-02	1151.	0.	0.	0.
3	PLEN	70.00	15.00	0.	62.00	34.55	0.	0.	0.

MODEL = PCOOK
FWDBCK

FLUINT SAMPLE PROBLEM 3 - PRESSURE COOKER

FLUID SUBMODEL NAME = COOKER ; FLUID NO. = 718

MAX TIME STEP = 0. ; LIMITING MESSAGE = (NO SOLUTION STEP TAKEN YET)
LAST TIME STEP = 0. VS. DTMAXF/DTMINF = 1.000000E+30 / 0. ; AVERAGE TIME STEP = 0.
PROBLEM TIME TIMEN = 0. VS. TIMEND = 8.333334E-03

TIE PARAMETER TABULATION FOR SUBMODEL COOKER

TIE	TYPE	UA	QTIE	LUMP	TEMP.	NODE	TEMP.	PATH 1	FRACT	PATH 2	FRACT
1	BTU	138.3	0.	1	250.0	WALL.1	257.0				

MODEL = PCOOK
FWDBCK

FLUINT SAMPLE PROBLEM 3 - PRESSURE COOKER

FLUID SUBMODEL NAME = COOKER ; FLUID NO. = 718

MAX TIME STEP = 0. ; LIMITING MESSAGE = (NO SOLUTION STEP TAKEN YET)
LAST TIME STEP = 0. VS. DTMAXF/DTMINF = 1.000000E+30 / 0. ; AVERAGE TIME STEP = 0.
PROBLEM TIME TIMEN = 0. VS. TIMEND = 8.333334E-03

PATH PARAMETER TABULATION FOR SUBMODEL COOKER

PATH	TYPE	LMP 1	LMP 2	DUP I	DUP J	STAT	XL UPSTRM	FLOWRATE	DELTA PRES	FLOW REGIMES OR REYNOLDS NO.
1	CTLVLV	1	2	1.0	1.0	VS	1.000	0.	15.08	
3	MFRSET	3	1	1.0	1.0	NORM	0.	0.	-4.1566E-02	

*** WARNING *** INACCURATE WITH CURRENT TIME STEP: LOWER DTIMEB OR RAISE DRLXCA
MODEL: WALL DIFF NODE 1

MODEL = PCOOK
FWDBCK

FLUINT SAMPLE PROBLEM 3 - PRESSURE COOKER

FLUID SUBMODEL NAME = COOKER ; FLUID NO. = 718

MAX TIME STEP = 8.568619E-04 ; LIMITING TANK = 1 REASON = QUALITY CHANGE LIMIT
LAST TIME STEP = 8.333330E-04 VS. DTMAXF/DTMINF = 1.000000E+30 / 0. ; AVERAGE TIME STEP = 3.876101E-04
PROBLEM TIME TIMEN = 8.333334E-03 VS. TIMEND = 8.333334E-03

LUMP PARAMETER TABULATION FOR SUBMODEL COOKER

LUMP	TYPE	TEMP	PRESSURE	QUALITY	DENSITY	ENTHALPY	HEAT RATE	MASS RATE	ENERGY RATE
1	TANK	222.5	3.273	4.4756E-02	0.9913	235.7	2496.	29.78	3525.
2	PLEN	212.0	-4.0171E-02	1.000	3.7257E-02	1151.	0.	0.	0.
3	PLEN	70.00	15.00	0.	62.00	34.55	0.	-29.78	-1029.

MODEL = PCOOK
FWDBCK

FLUINT SAMPLE PROBLEM 3 - PRESSURE COOKER

FLUID SUBMODEL NAME = COOKER ; FLUID NO. = 718

MAX TIME STEP = 8.568619E-04 ; LIMITING TANK = 1 REASON = QUALITY CHANGE LIMIT
LAST TIME STEP = 8.333330E-04 VS. DTMAXF/DTMINF = 1.000000E+30 / 0. ; AVERAGE TIME STEP = 3.876101E-04
PROBLEM TIME TIMEN = 8.333334E-03 VS. TIMEND = 8.333334E-03

TIE PARAMETER TABULATION FOR SUBMODEL COOKER

TIE	TYPE	UA	QTIE	LUMP	TEMP.	NODE	TEMP.	PATH 1	FRACT	PATH 2	FRACT
1	BTU	226.7	2496.	1	222.5	WALL.1	233.5				

MODEL = PCOOK
FWDBCK

FLUINT SAMPLE PROBLEM 3 - PRESSURE COOKER

FLUID SUBMODEL NAME = COOKER ; FLUID NO. = 718

MAX TIME STEP = 8.568619E-04 ; LIMITING TANK = 1 REASON = QUALITY CHANGE LIMIT
LAST TIME STEP = 8.333330E-04 VS. DTMAXF/DTMINF = 1.000000E+30 / 0. ; AVERAGE TIME STEP = 3.876101E-04
PROBLEM TIME TIMEN = 8.333334E-03 VS. TIMEND = 8.333334E-03

PATE PARAMETER TABULATION FOR SUBMODEL COOKER

PATH	TYPE	LMP 1	LMP 2	DUP I	DUP J	STAT	XL UPSTRM	FLOWRATE	DELTA PRES	FLOW REGIMES OR REYNOLDS NO.
1	CTLVLV	1	2	1.0	1.0	VS	1.000	0.	3.313	
3	MFRSET	3	1	1.0	1.0	NORM	0.	29.78	11.73	

SUBMODEL NAME = WALL

ARLXCA =	1.000000E-02	ARLXCC =	0.	ATMPCA =	1.000000E+30	ATMPCC =	0.	BACKUP =	0.
CSGFAC =	0.	CSGMAX =	0.	CSGMIN =	0.	DRLXCA =	1.000000E-02	DRLXCC =	0.
DTIMEB =	1.000000E+30	DTIMEI =	0.	DTIMEL =	0.	DTIMEU =	0.	DTMPCA =	1.000000E+30
DTMPCC =	0.	EBALNA =	0.	EBALNC =	0.	EBALSA =	1.000000E-02	EBALSC =	0.
ESUMIS =	0.	ESUMOS =	0.	EXTLIM =	50.0000	ITHOLD =	10	NARLXN =	0
NATMPN =	0	NCGMAN =	0	NCSGMN =	0	NDRLXN =	0	NDTMPN =	0
NEBALN =	0	NLOOPT =	100	ITEROT =	0	ITERXT =	3	OPEITR =	0
OUTPUT =	1000.00								

GLOBAL CONTROL CONSTANTS

ABSZRO =	-459.600	SIGMA =	1.00000	TIMEM =	8.333334E-03	TIMEN =	8.333334E-03	TIMEND =	1.00000
TIMEO =	8.333334E-03								

MODEL = PCOOK
FWDBCK

FLUINT SAMPLE PROBLEM 3 - PRESSURE COOKER

FLUID SUBMODEL NAME = COOKER ; FLUID NO. = 718

MAX TIME STEP = 8.272435E-04 ; LIMITING TANK = 1 REASON = ABS. PRESSURE CHANGE LIMIT
LAST TIME STEP = 9.313226E-08 VS. DTMAXF/DTMINF = 2.777778E-04 / 0. ; AVERAGE TIME STEP = 2.500091E-04
PROBLEM TIME TIMEN = 6.166662E-02 VS. TIMEND = 6.166662E-02

LUMP PARAMETER TABULATION FOR SUBMODEL COOKER

LUMP	TYPE	TEMP	PRESSURE	QUALITY	DENSITY	ENTHALPY	HEAT RATE	MASS RATE	ENERGY RATE
1	TANK	249.1	14.57	7.3481E-02	0.9543	289.3	975.4	-1.929	-1270.
2	PLEN	212.0	-4.0171E-02	1.000	3.7257E-02	1151.	0.	1.929	2245.
3	PLEN	70.00	15.00	0.	62.00	34.55	0.	0.	0.

MODEL = PCOOK
FWDBCK

FLUINT SAMPLE PROBLEM 3 - PRESSURE COOKER

FLUID SUBMODEL NAME = COOKER ; FLUID NO. = 718

MAX TIME STEP = 8.272435E-04 ; LIMITING TANK = 1 REASON = ABS. PRESSURE CHANGE LIMIT
LAST TIME STEP = 9.313226E-08 VS. DTMAXF/DTMINF = 2.777778E-04 / 0. ; AVERAGE TIME STEP = 2.500091E-04
PROBLEM TIME TIMEN = 6.166662E-02 VS. TIMEND = 6.166662E-02

TIE PARAMETER TABULATION FOR SUBMODEL COOKER

TIE	TYPE	UA	QTIE	LUMP	TEMP.	NODE	TEMP.	PATH 1	FRACT	PATH 2	FRACT
1	BTU	157.2	975.4	1	249.1	WALL.1	256.6				

MODEL = PCOOK
FWDBCK

FLUINT SAMPLE PROBLEM 3 - PRESSURE COOKER

FLUID SUBMODEL NAME = COOKER ; FLUID NO. = 718

MAX TIME STEP = 8.272435E-04 ; LIMITING TANK = 1 REASON = ABS. PRESSURE CHANGE LIMIT
LAST TIME STEP = 9.313226E-08 VS. DTMAXF/DTMINF = 2.777778E-04 / 0. ; AVERAGE TIME STEP = 2.500091E-04
PROBLEM TIME TIMEN = 6.166662E-02 VS. TIMEND = 6.166662E-02

PATH PARAMETER TABULATION FOR SUBMODEL COOKER

PATH	TYPE	LMP 1	LMP 2	DUP I	DUP J	STAT	XL UPSTRM	FLOWRATE	DELTA PRES	FLOW REGIMES OR REYNOLDS NO.
1	CTLVLV	1	2	1.0	1.0	VS	1.000	1.929	14.61	
3	MFRSET	3	1	1.0	1.0	NORM	0.	0.	0.4314	

PROBLEM D: SIMPLE TWO-PHASE PUMPED LOOP

This problem is the generic and artificial sample problem used in Section 3.11.3 of main volume to illustrate input formats. In addition to demonstrating the input formats of a wide variety of options, this problem was chosen to demonstrate the following FLUINT features:

1. user-described simplified two-phase fluids (7000 series)
2. simple capillary device
3. heat exchanger macros and forced convection ties
4. phase suction operation
5. the modeling of simple pumps
6. the use of a plenum to control pressure at one point in the loop
7. restart options
8. spatial acceleration effects

D.1 Problem Description

A simple ammonia two-phase circulation loop has been proposed as a candidate for the thermal management subsystem of a spacecraft. The project is in the conceptual design phase. The proposed design is shown in Figure D–1. The object of this analysis is to help decide whether such a system would best meet the mission requirements. A model is to be made that can be used in both steady-state and simple transient runs.

Heat is added to the loop via an evaporative coldplate of undetermined design which cools an electronics package of known power and efficiency. The waste heat load is estimated to be 400 W. The loop rejects this load in a 5 meter condensing heat exchanger connected to a very light-weight spacecraft radiator. The distance between the cold plate and the radiator will be about two meters, bringing the total loop length to 2*2.+5.0 = 9 meters. Most of the piping has a proposed inner diameter of 8 mm.

Since this is a preliminary design, no pump has been selected. However, a flow rate was estimated from the requirements to be 0.5 gm/s of liquid ammonia, and preliminary estimates of pump efficiencies were used to calculate the amount of heat added to the fluid by the pump as 15W.

An accumulator is needed to dampen pressure fluctuations and control the set-point temperature. This accumulator, whose design is unknown, will be placed at the outlet of the condenser (before the pump). It will be connected to the main loop by a small line.

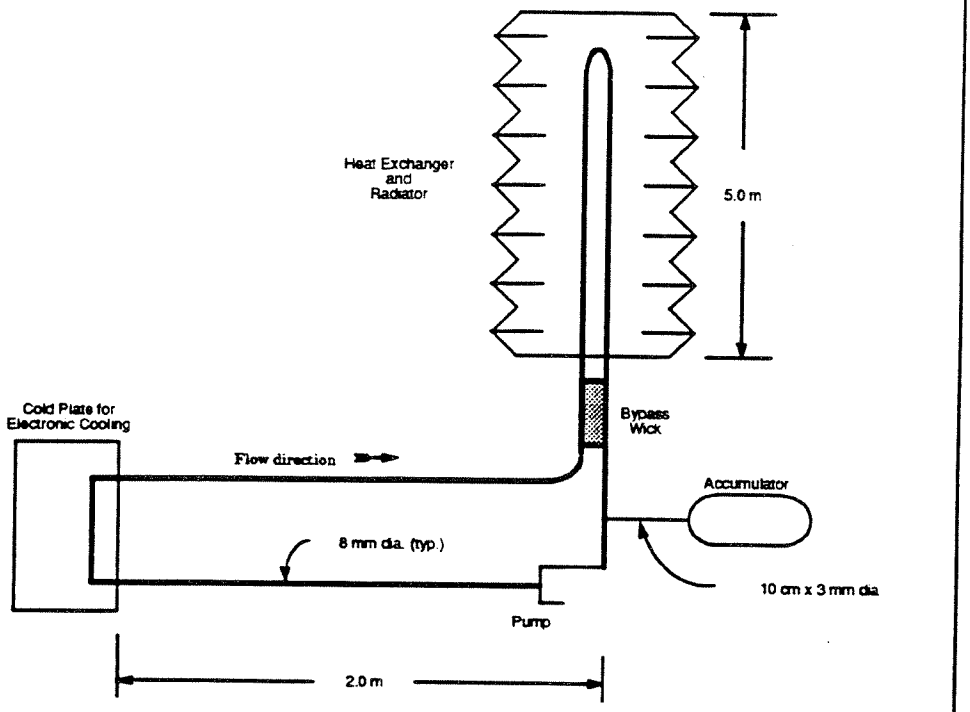


Figure D–1 Two-Phase Transport Loop with Capillary Bypass

The pump is purposely oversized in this system to assure that liquid ammonia is always present in the coldplate. The flow out of the coldplate and into the condenser will therefore be two-phase. The designer of the system decided to evaluate a new concept to minimize the amount of liquid being recirculated through the condenser. It is desired to place a rather thick (1 cm.) wick between the inlet and outlet of the condenser. The purpose of this wick is to absorb excess liquid in the exhaust line and allow this liquid to bypass the condenser, maximizing the vapor flow to condenser. The net resistance of the wick must be carefully matched against the resistance of the condenser to avoid depriming the wick or flooding the condenser excessively.*

The system parameters are summarized below:

Transport Piping:
- Total length ..... 2 m
- Inner diameter ..... 8 mm

Evaporator:
- Power ..... 400 W

Condenser Piping:
- Total length ..... 5 m
- Inner diameter ..... 8 mm

Radiator
- Total area ..... 3.75 m^2
- Emissivity 0.9
- Total mass (aluminum) ..... 5.0 kg
- Baseline Sink Temp. 250K

* No one, to the author's knowledge, has ever proposed such a concept. This sample is a contrivance for illustration purposes; the design should not be taken seriously.

Accumulator:	
Set-point	300K
Accumulator Piping:	
Length	0.1 m
Inner diameter	3 mm
Bypass Wick:	
Pore size (pumping radius)	16.5 microns
Permeability	2.6E-13 m^2
Flow area	0.02 m^2
Thickness	1 cm
Pump (ideal):	
Flowrate	0.5 gm/s
Cooling requirements	15 W

D.2 A FLUINT Model

The first modeling decision is to use an MFRSET connector for the pump. This will maintain the prescribed mass flowrate (which may be altered within user logic blocks if needed) independent of the pressure rise or fall or fluid state. The resulting pressure rise might be used later to resize the line diameter. If, on the other hand, the desired pressure difference were known but the flowrate could vary, a short STUBE with an HC equal to the desired pressure difference could be used as the pump. No extra losses will be attributed for bends or fittings since there is insufficient definition of the conceptual hardware. Pump-generated heat will be added to the fluid via the QDOT value of the lump at the pump outlet.

The second modeling decision is to use a plenum for the accumulator, which is equivalent to assuming ideal operation. The plenum will be connected to the system via an STUBE connector instead of a tube since the line is so short and the flowrates will be nearly zero*. In fact, if the line had been much shorter or had a larger diameter, a alternate decision would have been to assume that there was no difference between the pressure in the accumulator and that of the main line. This could be accomplished by eliminating the small path between the line and accumulator, and making the accumulator a "flow-through" device as an approximation. However, placing a plenum directly in the path of the flow in effect opens the loop—changes in the condenser will not be felt by the evaporator, and the plenum could become a significant source or sink of energy as well as mass.

The third decision is to use only one lump for the evaporator since the power input is known and no specific design exists as yet for the electronic coldplate. (In a two-phase spacecraft system, a capillary-assist device is likely and may require a different modeling approach.)

Fourth, it is decided to use five lumps for the heat transfer modeling of the radiator because of gradients in the radiator and the need to resolve the condensation process—flow regimes and qualities will change along the line. The first lump will contain mostly vapor, while the last lump in the condenser will contain subcooled liquid under normal conditions. Centered discretizing will be used to smooth the large property and heat flux gradients inherent in condensing two-phase flow. A duct macrocommand will be used to include pressure recovery due to axial velocity gradients.

* Flowrates in tubes can only change by a percentage of the existing flowrate during a solution interval. Tubes with flowrates hovering near zero will take many intervals to resolve.

The heat sink (radiator) will be modeled by a thermal network called RAD. If the radiator were isothermal, all five fluid lumps could tie to a single boundary node. In this case, however, the radiation exchange and the thermal capacitance of the radiator will be included in a first-order approach. Nodes will be generated to correspond to the five condenser segments, and these will be linked to an effective sink temperature (boundary node) via radiation conductors. These nodes will be linked to the junctions via HTNC ties. Conduction between nodes is neglected, and radiative fin efficiencies are assumed to be unity. Lower efficiencies can easily be multiplied by the radiator area and emissivity: $\varepsilon\eta A$.

The choice for the bypass wick should be clear: a CAPIL connector will be placed between the lumps representing the condenser inlet and outlet. This will have the effect of drying out the upstream lump (and hence the flow into the condenser), allowing only saturated liquid to pass into the condenser outlet as long as the capillary pressure limit is not exceeded.

Because of the preliminary nature of the model (i.e., liquid inventories and flow lag times are not important), junctions are used in the condensing heat exchanger. The volume contained in the condenser is divided evenly between two tanks located at the inlet and outlet of the condenser.

Because of nonnegligible volume, the evaporator lump is chosen to be a tank. Actually, in addition to illustration purposes, a tank is used to *help break up a closed loop of nonadiabatic junctions*. Such junction loops are illegal for the same reasons thermal submodels with only arithmetic nodes are illegal—there is no place for net energy imbalances to go.

The previous modeling decisions force the allocation of paths. Six paths in series are needed for the heat exchanger (and may be generated simultaneously with a macro), and two more are needed to connect the evaporator lump to the condenser and the pump. Because of their length and for illustration purposes, these last two are chosen to be tubes, but the rest can be approximated as STUBE connectors. If the condenser had been built with tanks, tubes would have been used instead.

Finally, a junction is added to close the loop.

An important remark regarding temporal resolution (i.e. the choice between tank vs. junction, tube vs. STUBE connector) is needed here. Strictly and mathematically, a network should be composed of either all time-dependent or all time-independent elements. However, such a restriction would severely handicap the analyst and eliminate the ability to make simplifying assumptions. The user must be cognizant of the inherent assumptions made when mixing tanks and tubes with junctions and STUBEs. In this example, the condenser can and will react instantly to any changes in boundary conditions, while most of the rest of the system cannot. Thus, the response lag in the rest of the system will tend to damp out condenser fluctuations, while the condenser will tend to drive the rest of the system to respond faster than it would if the condenser had been modeled with tanks and tubes. Modeling the condenser with tanks and tubes, however, would require smaller time steps to resolve internal condenser oscillations that are not considered relevant to the problem. The combination of tubes and junctions is acceptable if desired, but the combination of STUBEs and tanks should avoided in two-phase lines because of rapid, artificial transients.

Similarly, the choice of a tank or junction to model the condenser inlet has major consequences on the dynamic behavior of the system because of the phase suction in a CAPIL connector. To illustrate this point, assume the resistance in the wick is too low, such that the flowrate through the CAPIL is too high for the available liquid. A tank would oscillate near a quality of

1.0, and the CAPIL would oscillate between flowing liquid and no flow as long as it stayed primed. If instead a junction were used, it would stay at a high quality (but less than 1.0), and the CAPIL would operate continuously at a flowrate less than the full liquid value, possibly extracting some vapor.

The model developed here is shown in Figure D–2, and summarized below:

Fluid submodel LOOP:

TUBE 1	Liquid feed line from pump to evaporator.
TANK 1	Evaporator, containing all of the volume from the pump outlet to the condenser inlet.
TUBE 2	Vapor or two-phase line from evaporator to condenser.
TANK 2	Condenser inlet, containing half of the condenser volume.
JUNCTIONS 10-50	Heat exchanger segments that divide up the condenser into five parts.
STUBEs 10-60	Line segments in the condenser, corresponding to the five divisions described above. #10 and #60 are half-length.
HTNC TIES 10-50	Tie LOOP.N invokes automatic convection heat transfer between junction LOOP.N and node RAD.N. Note centered differencing and area fractions mean #10 includes all of STUBE #10 and 1/2 of STUBE #20, etc.
TANK 3	Tee at the outlet of the condenser that connects the condenser, the bypass, the line to the accumulator, and the pump. Contains the second half of the condenser volume.
CAPIL 3	Wicked bypass connecting condenser inlet and outlet
STUBE 100	Small line to link the accumulator into the system.
PLENUM 100	Accumulator.
MFRSET 200	Idealized pump (constant mass flowrate).
JUNCTION 200	Between pump and delivery line (TUBE 1).

Thermal submodel RAD:

NODES 10-50	Diffusion nodes, each modeling 1/5 of the radiator.
NODE 1	Boundary node, representing effective radiation sink.

Finally, *strictly for demonstration purposes,* a user-supplied description of two-phase ammonia will be used instead of the standard library fluid 717. This will allow comparisons between the two descriptions. While the user is encouraged to replace library fluids with RAPPR-generated or tabular simplified descriptions where possible for increased speed, a 7000 series description is less accurate and no faster than a library description. Refer to Sample Problem F for a more realistic application of 7000 series fluids.

D.3 The Input File

The input file is listed immediately following the last section.

OPTIONS DATA—A user file (USER1) is named. This file will contain a summary listing of the transient event. A restart output (RSO) file is also named to contain snapshots of the system for plotting or for future use. In this case, the results of the steady-state analyses are saved by the routine RESAVE. This file would then be named in a later run as an RSI (restart input) file, and the data would be accessed with a call to RESTAR.

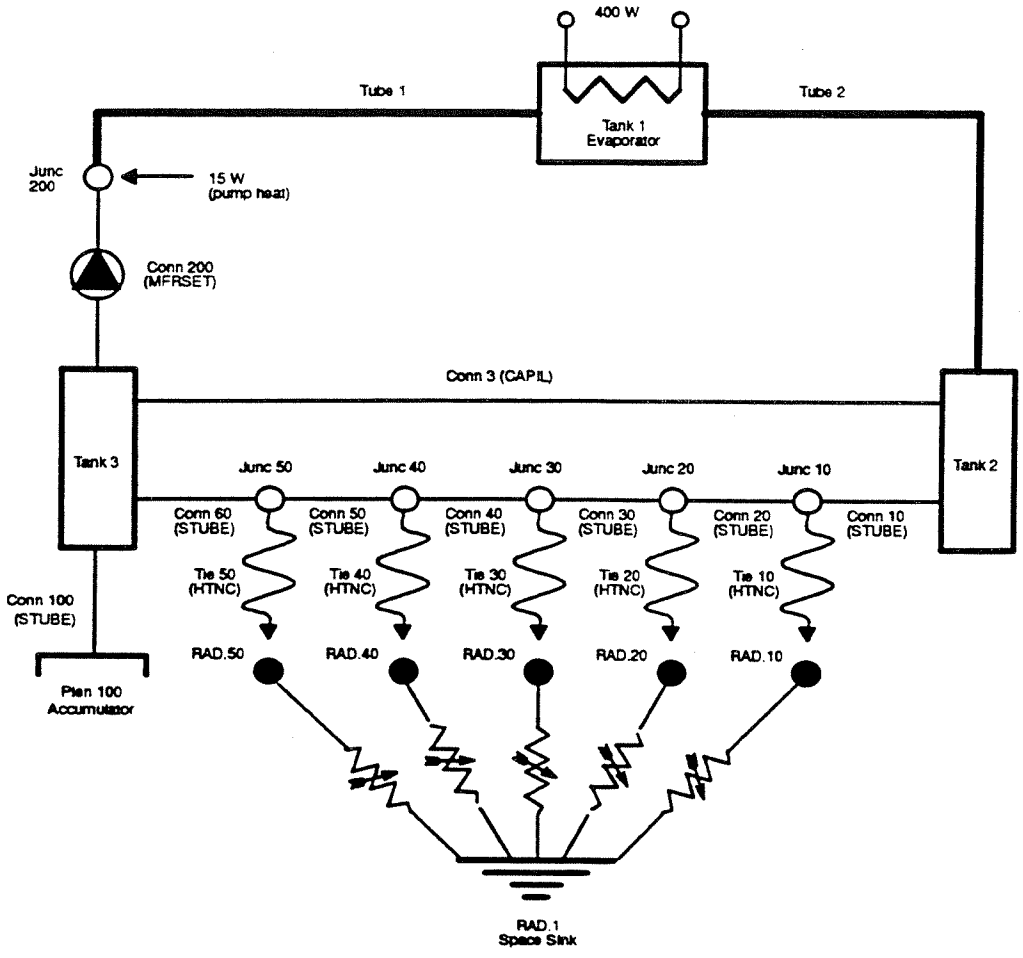


Figure D-2 Schematic of Pumped Loop Model

CONTROL DATA—Metric units are chosen, with temperature in Kelvin. The Stefan-Boltzmann constant is input here rather than inside CONDUCTOR DATA. NLOOPS is required and is set to 50. The values of OUTPTF and OUTPUT are required. OUTPUT is given here for all submodels as a large value since only fluid OUTPUT CALLS will be used. OUTPTF is set later in a fluid submodel-specific block to be 10 seconds, and RSMAXF (the maximum artificial time step or relaxation step in STDSTL) is also set to that value.

FPROP DATA—Inside the INCLUDEd file, an alternate, simplified, two-phase description of ammonia is named as fluid 7717. The description is input in English units, with temperature in degrees F. As long as these units are explicitly stated on the HEADER card, the program will perform the necessary unit conversions needed to comply with the unit set of the master network. Thus, this description can be reused in any model.

The required values for the gas constant and the critical point are entered. Further temperature and pressure limits are explicitly stated. The lower temperature limit is the freezing point; the upper pressure limit cuts off the upper portion of the dome where the assumptions inherent in a 7000 series fluid break down.

Of the remaining property values, all are input in array form except the surface tension, which is input as a value and temperature coefficient. The value corresponds to the temperature input as TREFL, which would also apply to other liquid properties that were not input as arrays. The array values for gas properties are input along the saturation line, except for CPG, which must be at zero or very low pressure. If significant superheat were expected and were important, these gas values (KG and VG) could have been entered along a relevant line of constant pressure.

Finally, the two required dome points are input at -105°F and 84°F. The first point is, as the manual recommends, just above freezing. The second point is taken well below the critical point near the expected range of temperatures. The heats of vaporization are input as the saturated vapor enthalpy minus the saturated liquid enthalpy, because the heat of vaporization was not listed in the reference used.

FLOW DATA—Only one submodel is used: a fluid submodel named LOOP. Refer to the previous section for descriptions of the model.

The working fluid is "7717", meaning a user-defined two-phase fluid described elsewhere in the input. The default state is given as liquid at 300K and 6 atmospheres. PLFACT is used to input pressures in atmospheres. This conversion applies to this FLOW DATA section only. Output pressures will still be listed in the default units of Pascals. The default initial flowrate is 0.5 gm/s, the default diameter is 8 mm, and the default length is 2 m. Note that IPDC=4 means that Friedel's frictional pressure drop correlation will be used in two-phase regions.

The quality of tank #1 is given as 0.1. This means that the default pressure will be ignored in favor of the saturation value at 300K. The volumes for tanks are calculated in the input file. This allows easy alteration of the volumes if lengths or diameters change, and provides some self-documentation for future users.

A centered HX macro has been used as an alternative to the 16 subblocks that would be required to specify the five condenser segments. One difference is that the initial conditions do not have to be linear (i.e., vary incrementally along the line) if each segment is named individually. More importantly, spatial accelerations are automatically calculated in a LINE or HX macro because the program knows that a duct is being modeled. Such effects would be ignored otherwise unless the user manipulated the AC values. Note that another macro (LINE) is available to generate duct models without ties. In other words, an HX macro is simply a LINE macro that also generates ties. LINE macros can be used for heat exchangers (with user-manipulated QDOT and/or Q arrays,) or for finer resolution of a long pipe. GENT,HXC macros may be used to add more ties around the duct periphery, perhaps to include wall gradients.

NODE DATA—Five diffusion nodes are generated with a GEN command, and one boundary node is defined.

CONDUCTOR DATA—Five radiation conductors are generated between the boundary node and the diffusion nodes in this submodel. SIGMA has been set in CONTROL DATA; otherwise it would be required here, perhaps as a FAC value.

FLOGIC 0—Two calculations are contained in this logic block. Both are minor points, performed mainly for demonstration purposes. First, the CHGLMP is used to maintain the plenum temperature near that of the condenser outlet temperature. This makes sure that small amounts of fluid entering the system from the plenum do not disturb the rest of the loop energetically. Second, the acceleration pressure drop in the evaporator is accounted for using the tube AC factors. When heat rates and/or tie UAs vary quickly (such as in steady-states) or with STUBEs, updating the AC term in a logic block can be destabilizing. LINE or HX macros should be used if possible when such terms are deemed important. (Radial blowing or suction is an exception.)

OUTPUT CALLS—This logic block defines the output to be produced at each output interval. Note that a compressed line of important data is written to the user file USER1 during transients (as dictated by the NSOL flag). The routine XTRACT is used to calculate the effective quality through the CAPIL connector. This quality would have been evident (allowing for flashing through the CAPIL) if there were a junction downstream of the CAPIL.

OPERATIONS DATA—The first order of business is to call PRPMAP to get a mapping of the user-defined fluid properties for verification purposes. PRPMAP is really only needed for debugging purposes when developing an alternate description. Here, it is used to compare the library description with the simple user description.

Second, a steady-state analysis is performed, with STDSTL being used to verify a FASTIC run. At this point in the execution, RESAVE is used to fill the RSO file with a snapshot of the network for future restarts. Third, a header is written to the USER1 file.

The remainder of this logic block contains the three parts of the transient event. Three sequential FORWRD calls with the sink temperature reset each time. Each run uses the last network state as its initial conditions. Otherwise, the histories would not be continuous in time.

D.4 Output Description

The user summary file and selected printings from the OUTPUT file are listed following the input file.

Inspection of the PRPMAP output reveals generally good agreement between the library fluid 717 and the simple user-defined fluid 7717. The points of disagreement should be noted. First, the critical densities are different. The 717 value is correct; the 7717 value is based on the Van der Waal equation of state, and underscores the fact that such fluids cannot be used near the critical point. Pressures should always be well below the critical pressure. Entropies and enthalpies are very different, but only because different reference points are used. Only *differences* in these properties are useful analytically, and the differences shown here are similar.

As noted in the manual, the property with the least accuracy is the specific heat for liquid. For the 7717 fluid, this value (as well as the specific heat of vapor at high pressures) is sometimes 50% in error. Because of the relative insignificance of sensible heat storage for two-phase systems, this apparently large error is perfectly acceptable. In fact, there are no appreciable differences in the answers when this sample problem is run with the library description of ammonia. *However, the user should avoid using a 7000 series fluid for single-phase analyses.* 8000

and 9000 series (or even 6000 series) are intended for that purpose. The 7000 series is intended to allow quick descriptions of alternate fluids for two-phase analyses (see Sample Problem F for a more realistic application of 7000 series fluids). The 6000 series is intended for more detailed or wider ranging descriptions, but is also used by RAPPR to generate limited-range but fast descriptions of library fluids.

With regard to the network behavior, FASTIC converges to a steady-state solution after 30 iterations. The subsequent STDSTL run confirms these answers. The full output file contains a warning that the CAPIL has "deprimed" because of excess pressure gradient. This deprime mechanism would allow vapor to pass through the wick because the bubble point or capillary limit has been exceeded. However, before convergence the pressure gradient drops to below the capillary limit, although no reprime message is produced because excess liquid at the inlet prevents formation of a steady meniscus. This does not mean that vapor is allowed to pass, as is the case if the pressure gradient exceeded the capillary limit. Rather, this means that the wick does not stop the flow of liquid, which falls to the lower pressure outlet. Under these conditions, tank #2 is drier than tank #1, but there is enough liquid to satisfy the flow through the wick. The higher quality flow enters the condenser.

As seen in the OUTPUT file, a restart record number is written as "50" after the STDSTL call. Using this record number in a call to RESTAR in future runs will return the system to the steady-state solution for further analyses. Unlike this example, in most real runs steady-states are expensive and should be saved for future use. As with parametric runs (RESPAR/SAVPAR), multiple states can be saved on any one file. *In order to restart, the network must be the same as it was when saved.* This means that the BUILD/BUILDF operations must be the same, and that no nodes, conductors, lumps, paths, user constants, arrays, etc., have been added or deleted. *In order to restart a fluid submodel, all flow data must be saved*; fluid submodels cannot be restarted with partial data as can thermal submodels.

Restart files may be used interchangeably with SAVE files. By adding a call to SAVE or RESAVE in the OUTPUT CALLS block during the transient, the user not only preserves the option of restarting the run at any saved time, he may interrogate the RSO or SAVE file using either DATA_ONLY or EXPLOT. This frees the user from having to decide beforehand which data is important to tabulate or plot. The only cost of such an option is the storage cost of the potentially huge files that are generated if large models are saved frequently.

During the transient events, the quality in the evaporator is nearly invariant because of the relative insignificance of sensible heating compared to vaporization. However, the flowrates in the condenser and bypass wick vary according to environmental conditions. A colder environment means a more flooded condenser. This, in turn, lowers the flow resistance through the condenser, diverting flow from the bypass. The opposite occurs during the hot transient. In fact, the inlet portion of the radiator becomes nearly as warm as the vapor during the hot transient; very little condensation occurs in the first segment. Note the drastic changes in condenser heat rates even though the remainder of the system does not react significantly during these short events.

The flowrates through the condenser and bypass nearly always sum to the system flowrate (Figure D–3). This does not have to be true: tank #2 can and does fill and empty, though gradually. It is very typical of two-phase tanks that the flowrates in and out do not equal each other for long periods of time. The net rate of mass storage in the tank is listed in the LMPTAB output, as well as the net rate of energy storage. Both of these terms should be nearly zero at steady-state. Nonzero terms can provide clues to the reasons for nonconvergence in STDSTL.

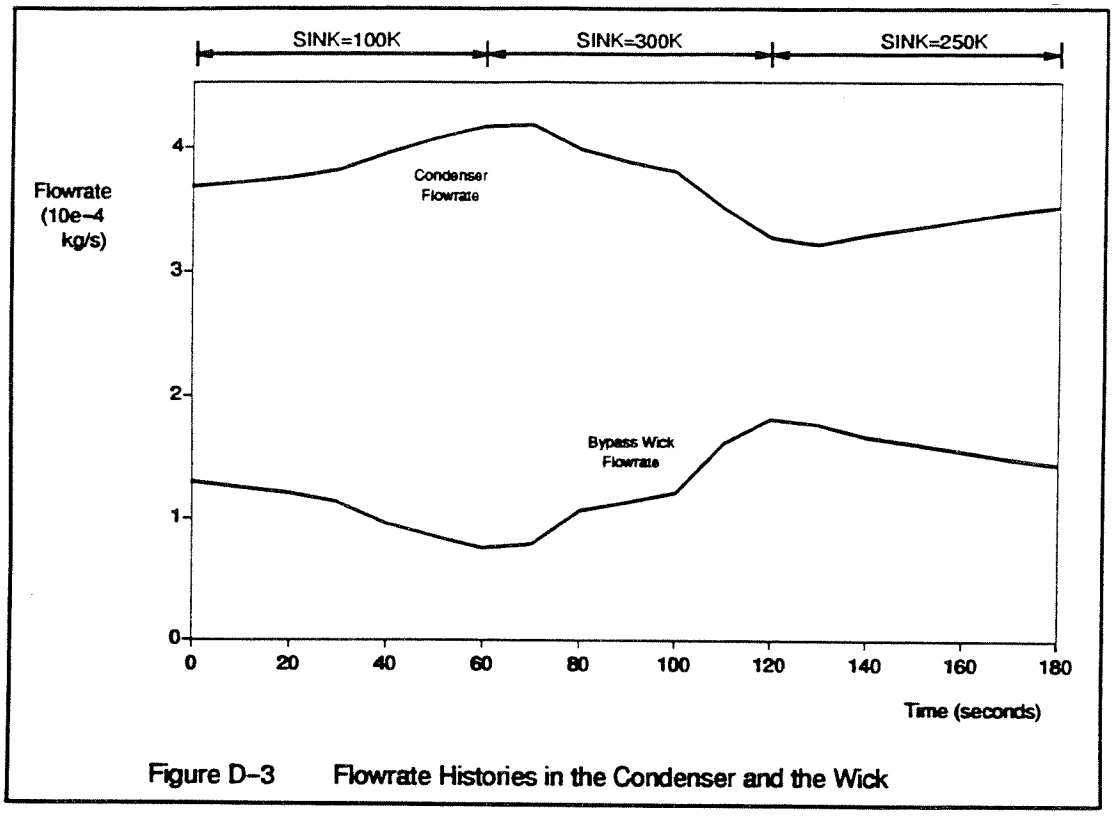


Figure D-3 Flowrate Histories in the Condenser and the Wick

Under different circumstances, the output file might reveal an occasional warning message regarding condenser junction temperatures. Such warnings are very common with tied junctions, and are inconsequential unless they are persistent. They are a residue of the fact that QDOTs are held constant over the transient solution interval (to conserve energy with thermal models) even though the flowrate though a junction can drop off during the solution for hydrodynamic reasons. Application of a QDOT to a junction with insufficient flowrate can quickly and temporarily drive the state of the junction to the property range limits of the working fluid. During subsequent solution steps, the QDOT is quickly readjusted to the new conditions. Similarly, temporary error conditions may exist on tied arithmetic nodes during transients; the causes and the consequences of such errors are analogous to the tied junction error messages.

Input File

```
HEADER OPTIONS DATA
TITLE FLUINT SAMPLE MODEL
C
C FLUINT SAMPLE MODEL INPUT DECK
C
C SEE USER'S MANUAL SECTION 3 FOR SCHEMATICS.
C SEE USER'S MANUAL SECTION 4 FOR MORE DISCUSSION.
C
C THIS FILE DESCRIBES THE FLUID SYSTEM TO THE PROGRAM,
C PERFORMS A STEADY-STATE SOLUTION, SAVE THE RESULTS IN A
C RESTART FILE FOR FUTURE USE, AND THEN PERFORMS A SIMPLE
C TRANSIENT IN WHICH THE RADIATOR SINK TEMPERATURE DROPS AT
C TIME ZERO FROM 250 K TO 100 K, RISES TO 300 K AT T=30s,
C THEN DROPS BACK TO 250K AT T=60s.
C
C NOTE THE SENSITIVITY TO CHANGES IN THE NODE TEMPS.
C THESE ARE ASSUMED TO BE INNER WALL TEMPERATURES BY FLUINT
C AND HENCE ARE ADJACENT TO THE FLUID!!
C
         MODEL    = SAMPLE
         OUTPUT   = fsample.out
         RSO      = fsample.rso
         USER1    = fsample.usr
C
HEADER CONTROL DATA,GLOBAL
         NLOOPS   = 50              $ 50 ITERATIONS FOR STEADY-STATE
         UID      = SI              $ MODEL UNITS ARE SI (K-m-kg-s)
         ABSZRO   = 0.0             $ TEMPERATURE UNITS ARE KELVIN
         OUTPUT   = 1000.0          $ LARGE VALUE TO SATISFY THERMAL
         SIGMA    = 5.67E-8         $ SB CONST IN METRIC UNITS
C                                   FORWRD REQ'MTS. OUTPTF USED INSTEAD.
HEADER CONTROL DATA,LOOP
         OUTPTF   = 10.0            $ OUTPUT EVERY 10 SECONDS
         RSMAXF   = 10.0            $ SET MAX SS TIME STEP TO BE SAFE
C
C FETCH USER-DEFINED AMMONIA INPUT FILE (7000 SERIES NAMED 7717)
C
INCLUDE fsample.inc
```

```
HEADER FLOW DATA,LOOP,FID=7717      $ AMMONIA FILLED "LOOP"
C
C DEFINE MODEL DEFAULTS
C
LU DEF, TL        = 300.0           $ DEFAULT TEMPERATURE IS 300 K
         PLFACT   = 101325.0        $ PL MULTIPLIER (EXPECT ATM)
         PL       = 6.0             $ 6 ATMOS IS DEFAULT PRESSURE
         XL       = 0.0             $ DEFAULT STATE IS LIQUID
PA DEF, FR        = 5.0E-4          $ FLOWRATE IS 0.5 GRAMS/SEC
         DH       = 0.008           $ DEFAULT DIAMETER IS 8 mm
         TLEN     = 2.0             $ DEFAULT DUCT LENGTH IS 2 METERS
         WRF      = 1.0E-6          $ ASSUME FAIRLY SMOOTH PIPE
         UPF      = 0.5             $ USE WEIGHTED AVG. OF UP/DOWNSTREAM
         IPDC     = 4               $ USE FRIEDEL'S CORRELATION
C
C START AT PUMP OUTLET, GO FLOWWISE AROUND LOOP
C
PA TUBE,1,200,1                    $ TUBE FROM PUMP TO EVAPORATOR
C
LU TANK,1                          $ EVAPORATOR CONTROL VOLUME
         VOL      = 4.0*0.008*0.008*0.25*3.1416
         QDOT     = 400.0
         XL       = 0.1             $ NOTE PL WILL BE P AT SATURATION
C
PA TUBE,2,1,2                      $ TUBE FROM EVAP TO SEPARATOR
C
LU TANK,2                          $ SEPARATOR AND UPPER 1/2 OF COND
         VOL      = 2.5*0.008*0.008*0.25*3.1416
         XL       = 0.1             $ NOTE PL WILL BE P AT SATURATION
C
C
C WICK A = 0.02, T = 1 cm.
C
PA CONN,3,2,3,     FR = 1.0E-4,    DEV = CAPIL
                   RC = 16.5E-6,   CFC = 2.6E-13*0.02/0.01
C
C START CONDENSER LINE (THE FOLLOWING 20 LINES WERE REPLACED BY
C AN M HX,C MACRO SUBBLOCK.  THIS INCLUDES DECELERATION OF VAPOR)
C USE CENTERED DIFFERENCING
C
CLU JUNC,10, XL   = 0.5            $
CLU JUNC,20, XL   = 0.3            $ NOTE QUALITY "GRADIENT"
CLU JUNC,30, XL   = 0.1            $
CLU JUNC,40, TL   = 290.0          $ DEFAULT XL=0.0
CLU JUNC,50, TL   = 270.0          $ SUBCOOLED SECTION
C
C NOW DESCRIBE STUBE CONNECTORS THAT CORRESPOND TO ABOVE JUNCS
C
```

```
CPA DEF, TLEN = 1.0
CPA CONN,10,3,10,          DEV = STUBE,      TLEN = 0.50
CPA CONN,20,10,20,         DEV = STUBE
CPA CONN,30,20,30,         DEV = STUBE
CPA CONN,40,30,40,         DEV = STUBE
CPA CONN,50,40,50,         DEV = STUBE
CPA CONN,60,50,3,          DEV = STUBE,      TLEN = 0.50
C
C SET HEAT TRANSFER TIES IN CONDENSER
C NOTE CENTER PATHS ARE SHARED AMONG TIES
C
CT HTNC,10,10,RAD.10,      10,1.0,  20,0.5
CT HTNC,20,20,RAD.20,      20,0.5,  30,0.5
CT HTNC,30,30,RAD.30,      30,0.5,  40,0.5
CT HTNC,40,40,RAD.40,      40,0.5,  50,0.5
CT HTNC,50,50,RAD.50,      50,0.5,  60,1.0
C
C THE PREVIOUS 20 OR SO LINES WERE REPLACED WITH:
C
M HX,1,C,10,10,10,RAD.10,2,3
         NSEG = 5,         PA = STUBE,       LU = JUNC
         NINC = 10,        LUINC = 10,       PAINC = 10, TIINC = 10
         TLENT = 5.0,      DHS = 0.008,      FR = 4.0E-4
         XL = 0.4,         XLINC = -0.1,     UPF = 0.5
C
C NOTE THAT INITIAL CONDITIONS ARE DIFFERENT
C AND THAT AN HX MACRO ADDS ACCELERATION FACTORS AUTOMATICALLY
C
C FINISH LOOP TO PUMP, THEN ADD ACCUMULATOR
C
LU DEF, TL        = 270.          $ RESET DEFAULT TEMP. TO 270 K
C
C THE FOLLOWING TANK CONTAINS THE SECOND HALF OF THE
C VOLUME OF THE CONDENSER.
C
LU TANK,3,        VOL = 2.5*0.008*0.008*0.25*3.1416
PA CONN,200,3,200, DEV = MFRSET     $ SIMPLEST PUMP MODEL
C                          NOTE THE SMFR EQUALS THE DEFAULT FR.
C
LU JUNC,200                         $ JUNC TO MATE PUMP TO EVAP. TUBE
         QDOT     = 15.0            $ 15 WATTS PUMP HEATING
PA CONN,100,3,100                   $ NOTE FLOW IS POS INTO ACCUM
         FR       = 0.0
         DEV      = STUBE
         TLEN     = 0.1
         DH       = 0.003
LU PLEN,100                         $ ACCUMULATOR
```

```
C
C DONE DESCRIBING MODEL, NOW DESCRIBE OPERATIONS AND OUTPUT
C
HEADER OUTPUT CALLS, LOOP
         CALL LMPTAB('LOOP')
         CALL TIETAB('LOOP')
         CALL PTHTAB('LOOP')
C
         IF(NSOL .LE. 1) RETURN
         CALL XTRACT(XTEST,'LOOP',3)
         WRITE(NUSER1,10)TIMEN,XL1,FR10,FR3,XL2,XTEST,TL1-TL50,PL2-PL3
F10      FORMAT(1X,1P,8(G12.4,3X))
C
C FOR SIMPLICITY THE RADIATOR MODEL CONSISTS ONLY OF THE
C CONDENSER PIPE WALL, MODELED BY FIVE DIFFUSION NODES
C THAT ARE IN TURN RADIATING TO A BOUNDARY NODE.  OTHER
C TRANSPORT BETWEEN NODES IS NEGLECTED, AND FIN EFFICIENCIES
C ARE UNITY.  EACH NODE HAS AN EMISSIVITY OF 0.9,
C AN AREA OF 0.75 M^2, A MASS OF 1.0 KG OF ALUMINUM AT CP=900
C
HEADER NODE DATA,RAD
         GEN 10,5,10,280.0,1.0*900.0          $ RADIATOR NODES
         -1,250.0,0.0                         $ SPACE BOUNDARY
HEADER CONDUCTOR DATA,RAD
         GEN -10,5,10,10,10,1,0,0.75*0.9      $ RADIATOR NODES
```

```
HEADER OPERATIONS DATA
C
C FIRST, DO A PROPERTY MAPPING OF FLUID 7717 AND THEN
C LIBRARY FLUID 717 FOR COMPARISON.  USE LOW LIMITS
C ON TEMP AND PRESSURE TO NARROW RANGE OF COMPARISON
C
         CALL PRPMAP(1.3D6,300.0,FI7717)
         CALL PRPMAP(1.3D6,300.0,FI717)
BUILDF SAMPLE,LOOP
BUILD SAMPLE,RAD
C
C PLENUM WILL GOVERN SYSTEM TEMPERATURE.  RESET PLENUM
C PRESSURE TO BE THAT OF EVAPORATOR (SAT AT 300K)
C
         CALL CHGLMP('LOOP',100,'PL',SNGL(LOOP.PL1),'XL')
C
C PERFORM STEADY STATE WITH FASTIC THEN STDSTL,
C THEN SAVE IT, RESET RADIATOR TEMPS AND DO TRANSIENT
C
C NOTE STEADY STATE IS DONE BY A FIRST CALL TO FASTIC THEN
C TO STDSTL.  THIS IS A GOOD STRATEGY FOR FAST CONVERGENCE.
C THE USER MIGHT HAVE CHANGED NLOOPS BETWEEN THESE CALL TO
C USE ONE ROUTINE IN FAVOR OF ANOTHER.
C
         CALL FASTIC
         CALL STDSTL
         CALL RESAVE('ALL')
C
C WRITE HEADER TO OUTPUT FILE
C
         WRITE(NUSER1,10)
FSTART
10       FORMAT(/'   TIME (SEC)',T19,'XL EVAP',8X,'FR COND',8X
     .          ,'FR BYPASS',6X,'X COND',9X,'X BYPASS',7X
     .            ,'SUBCOOL',8X,'DP WICK,COND'/)
11       FORMAT('  RADIATOR SINK TEMP. RESET TO ',F6.1,' K')
FSTOP
DEFMOD RAD
C
C 0 to 60 sec, RADIATOR TEMP DROPS
C
         TIMEND   = 60.0
         T1       = 100.0
         WRITE(NUSER1,11)T1
         CALL FORWRD
C
C 60 to 120 sec, RADIATOR TEMP RISES
C
```

```
          TIMEND    = 120.0
          T1        = 300.0
          WRITE(NUSER1,11)T1
          CALL FORWRD
C
C 120 to 180 sec, RADIATOR TEMP RETURNS TO NORMAL
C
          TIMEND    = 180.0
          T1        = 250.0
          WRITE(NUSER1,11)T1
          CALL FORWRD
C
HEADER FLOGIC 0, LOOP
C
C THIS BLOCK CHANGES THE ACCUMULATOR TEMPERATURE
C TO KEEP IT FROM LAGGING BEHIND.  THIS IS NOT NECESSARY,
C BUT OTHERWISE THE PLENUM TEMPERATURE WOULD BE VERY
C DIFFERENT FROM THE REST OF THE LOOP AFTER THE STEADY STATE.
C
C NOTE THAT THE STATEMENT "TL100 = TL3" WOULD BE AGAINST THE RULES
C
C UPDATE PLENUM TEMPERATURE WITH THAT OF TANK 3
C AND MAKE SURE IT STAYS LIQUID
C
          CALL CHGLMP('LOOP',100,'TL',TL3,'PL')
          CALL CHGLMP('LOOP',100,'XL',0.0,'PL')
C
C THE NEXT BLOCK ACCOUNTS FOR THE FACT THAT THE FLUID
C BOILED IN THE EVAPORATOR WAS ACCELERATED, ADDING IN
C AN ADDITIONAL PRESSURE DROP.  THE AC FACTOR IS USED
C AND IS DIVIDED UP BETWEEN THE TWO TUBES.  WITHOUT THESE
C TERMS, ONLY FRICTIONAL LOSSES WOULD BE ACCOUNTED FOR.
C (THIS WOULD BE ACCOUNTED FOR AUTOMATICALLY IN A CENTERED,
C ONE SEGMENT LINE MACRO WITH TLENT=4.0.  FOR A VARIETY
C OF REASONS, A LINE MACRO WOULD BE BETTER IN OTHER MODELS)
C
          ATEST     = (1.0/DL3 - 1.0/DL1)/(AF1)**2
          AC1       = ATEST*0.5
          AC2       = AC1
C
END OF DATA
```

```
C
C THIS IS THE FPROP INCLUDE FILE FOR THE FSAMPLE CASE
C (FLUINT SAMPLE PROBLEM D)
C
HEADER FPROP DATA,7717,ENG,-459.6
C
C FAIRLY SIMPLE AMMONIA IN ENGLISH UNITS, DEGREES F
C
         RGAS     = 1545.33/17.03          $ GAS CONSTANT
         TMIN     = -107.9                 $ MIN TEMP ALLOWED
         TGMAX    = 180.0                  $ MAX TEMP ALLOWED
         PGMAX    = 200.0                  $ MAX PRESS ALLOWED
         TCRIT    = 271.4                  $ CRITICAL TEMP
         PCRIT    = 1657.0                 $ CRITICAL PRESSURE
C
         ST       = 23.4E-3/14.5932        $ SURFACE TENSION
         STTC     = -2.4E-4/14.5932/1.8    $ ST TEMP COEF.
         TREFL    = 51.6                   $ ST IS A LIQ PROPERTY!
C
C LIQ. DENSITY, VISCOSITY, CONDUCTIVITY, AND CP (GAS, ZERO P) ARRAYS.
C THESE COULD ALSO HAVE BEEN INPUT AS (VALUE,TEMP COEF.) PAIRS ABOVE
C
AT,DL,   -105.0,45.71,     0.0,41.34,      84.0,37.26,  125.0,34.96
AT,VL,   -20.0,0.629,      40.0,0.437,     80.0,0.341,  180.0,0.190
AT,VG,   -20.0,0.0227,     80.0,0.0277,    180.0,0.0340
AT,KL,   -100.0,0.41,      -20.0,0.35,     40.0,0.306,  80.0,0.276
         180.0,0.201
AT,KG,   -20.0,0.011,      80.0,0.016,     180.0,0.029
AT,CPG,  -60.0,0.477,      20.0,0.489,     80.0,0.500,  180.0,0.521
C
C MANDATORY DOME PROPERTIES
C
AT,DOME, -105.0,  0.996,   570.3+68.5      $ NEAR FREEZING
         84.0,    163.7,   631.3-136.6     $ NEAR OPER POINT
```

Processor Output

TIME (SEC)	XL EVAP	FR COND	FR BYPASS	X COND	X BYPASS	SUBCOOL	DP WICK,COND
RADIATOR SINK TEMP. RESET TO 100.0 K							
0.	0.5660	3.6933E-04	1.3099E-04	0.7663	0.	37.83	59.10
10.00	0.5658	3.7236E-04	1.2589E-04	0.7595	0.	38.65	56.80
20.00	0.5649	3.7646E-04	1.2143E-04	0.7495	0.	39.50	54.79
30.00	0.5633	3.8254E-04	1.1423E-04	0.7351	0.	41.07	51.54
40.00	0.5613	3.9646E-04	9.6792E-05	0.7064	0.	43.04	43.67
50.00	0.5586	4.0816E-04	8.6217E-05	0.6825	0.	44.65	38.90
60.00	0.5557	4.1776E-04	7.6629E-05	0.6634	0.	46.49	34.58
RADIATOR SINK TEMP. RESET TO 300.0 K							
60.00	0.5557	4.1776E-04	7.6629E-05	0.6634	0.	46.49	34.58
70.00	0.5528	4.1930E-04	7.9486E-05	0.6574	0.	46.24	35.86
80.00	0.5509	3.9996E-04	1.0683E-04	0.6874	0.	43.89	48.21
90.00	0.5502	3.8958E-04	1.1368E-04	0.7056	0.	42.40	51.29
100.0	0.5506	3.8171E-04	1.2134E-04	0.7210	0.	41.60	54.75
110.0	0.5518	3.5286E-04	1.6120E-04	0.7823	0.	38.14	72.74
120.0	0.5542	3.2879E-04	1.8163E-04	0.8443	0.	35.76	81.95
RADIATOR SINK TEMP. RESET TO 250.0 K							
120.0	0.5542	3.2879E-04	1.8163E-04	0.8443	0.	35.76	81.95
130.0	0.5567	3.2274E-04	1.7734E-04	0.8646	0.	35.02	80.02
140.0	0.5587	3.3085E-04	1.6711E-04	0.8458	0.	34.71	75.40
150.0	0.5601	3.3705E-04	1.6172E-04	0.8317	0.	34.45	72.97
160.0	0.5612	3.4303E-04	1.5566E-04	0.8187	0.	34.44	70.24
170.0	0.5621	3.4885E-04	1.4974E-04	0.8061	0.	34.49	67.57
180.0	0.5628	3.5390E-04	1.4490E-04	0.7956	0.	34.53	65.38

SYSTEMS IMPROVED NUMERICAL DIFFERENCING ANALYZER '85 (SINDA '85) PAGE 1

MODEL = SAMPLE FLUINT SAMPLE MODEL
PRPMAP

PROPERTY MAPPING OF USER DESCRIBED SIMPLIFIED TWO-PHASE 7717

MINIMUM PRESSURE:	6043.9		
MAXIMUM GAS PRESSURE:	1.37895E+06	PRES. LIMIT FOR PRPMAP:	1.06667E+06
MINIMUM TEMPERATURE:	195.39		
MAXIMUM GAS TEMPERATURE:	355.33	GAS TEMP. LIMIT FOR PRPMAP:	300.00
MAXIMUM LIQUID TEMPERATURE:	309.40	LIQ. TEMP. LIMIT FOR PRPMAP:	300.00
CRITICAL TEMPERATURE:	406.11		
CRITICAL PRESSURE:	1.14246E+07		
CRITICAL DENSITY:	152.56		

MODEL = SAMPLE
PRPMAP

FLUINT SAMPLE MODEL

SATURATION DOME DATA FOR USER DESCRIBED SIMPLIFIED TWO-PHASE 7717

TEMP.	SAT. PRESS.	DERIV. DPDT	HEAT OF VAP.	VAPOR SP. VOL.
195.39	6043.9	481.09	1.48421E+06	15.791
198.00	7423.8	576.56	1.48688E+06	13.026
200.62	9072.9	687.33	1.48862E+06	10.797
203.23	11034.	815.14	1.48944E+06	8.9920
205.85	13353.	961.81	1.48939E+06	7.5240
208.47	16083.	1129.2	1.48848E+06	6.3246
211.08	19279.	1319.3	1.48675E+06	5.3404
213.70	23004.	1533.9	1.48423E+06	4.5294
216.31	27326.	1775.1	1.48094E+06	3.8582
218.93	32314.	2044.8	1.47692E+06	3.3005
221.54	38048.	2345.0	1.47219E+06	2.8352
224.16	44608.	2677.5	1.46679E+06	2.4454
226.77	52082.	3044.0	1.46075E+06	2.1176
229.39	60561.	3446.3	1.45409E+06	1.8408
232.00	70141.	3886.0	1.44684E+06	1.6063
234.62	80921.	4364.5	1.43904E+06	1.4068
237.23	93004.	4883.0	1.43071E+06	1.2365
239.85	1.06498E+05	5442.9	1.42189E+06	1.0907
242.46	1.21510E+05	6044.9	1.41261E+06	0.96527
245.08	1.38153E+05	6689.9	1.40290E+06	0.85714
247.69	1.56541E+05	7378.7	1.39279E+06	0.76356
250.31	1.76786E+05	8111.4	1.38232E+06	0.68232
252.93	1.99006E+05	8888.6	1.37152E+06	0.61157
255.54	2.23317E+05	9710.3	1.36042E+06	0.54976
258.16	2.49836E+05	10576.	1.34905E+06	0.49561
260.77	2.78677E+05	11487.	1.33745E+06	0.44802
263.39	3.09957E+05	12442.	1.32566E+06	0.40608
266.00	3.43791E+05	13440.	1.31371E+06	0.36901
268.62	3.80292E+05	14481.	1.30162E+06	0.33616
271.23	4.19574E+05	15566.	1.28945E+06	0.30697
273.85	4.61749E+05	16693.	1.27722E+06	0.28097
276.46	5.06925E+05	17862.	1.26496E+06	0.25774
279.08	5.55215E+05	19073.	1.25272E+06	0.23693
281.69	6.06725E+05	20326.	1.24052E+06	0.21826
284.31	6.61565E+05	21619.	1.22839E+06	0.20146
286.92	7.19843E+05	22955.	1.21638E+06	0.18630
289.54	7.81668E+05	24332.	1.20451E+06	0.17260
292.15	8.47150E+05	25751.	1.19281E+06	0.16018
294.77	9.16400E+05	27214.	1.18131E+06	0.14891
297.38	9.89533E+05	28721.	1.17005E+06	0.13865
300.00	1.06667E+06	30273.	1.15904E+06	0.12929

MODEL = SAMPLE
PRPMAP

FLUINT SAMPLE MODEL

SATURATED LIQUID PROPERTIES FOR USER DESCRIBED SIMPLIFIED TWO-PHASE 7717

TEMP.	DENSITY	SP. HEAT	VISCOSITY	CONDUCT.	ENTHALPY	SURFACE TENS.	SOUND SPEED
195.39	734.14	889.41	3.76249E-04	0.71997	-1.12160E+06	4.46666E-02	2688.3
198.00	731.00	970.92	3.70023E-04	0.71385	-1.11917E+06	4.40389E-02	3271.6
200.62	727.86	1330.1	3.63796E-04	0.70774	-1.11579E+06	4.34113E-02	4307.7
203.23	724.72	1678.2	3.57570E-04	0.70163	-1.11150E+06	4.27836E-02	2671.9
205.85	721.58	2012.3	3.51343E-04	0.69552	-1.10633E-06	4.21559E-02	2249.1
208.47	718.45	2336.7	3.45117E-04	0.68941	-1.10031E+06	4.15283E-02	2038.5
211.08	715.31	2648.3	3.38890E-04	0.68330	-1.09347E+06	4.09006E-02	1908.7
213.70	712.17	2948.0	3.32664E-04	0.67719	-1.08585E+06	4.02729E-02	1817.9
216.31	709.03	3236.7	3.26437E-04	0.67107	-1.07746E+06	3.96453E-02	1748.9
218.93	705.89	3513.1	3.20211E-04	0.66496	-1.06834E+06	3.90176E-02	1693.6
221.54	702.75	3777.9	3.13985E-04	0.65885	-1.05853E+06	3.83899E-02	1647.4
224.16	699.62	4030.4	3.07758E-04	0.65274	-1.04806E+06	3.77623E-02	1607.6
226.77	696.48	4272.1	3.01532E-04	0.64663	-1.03694E+06	3.71346E-02	1572.3
229.39	693.34	4499.8	2.95305E-04	0.64052	-1.02523E+06	3.65069E-02	1540.4
232.00	690.20	4716.1	2.89079E-04	0.63441	-1.01294E+06	3.58793E-02	1511.2
234.62	687.06	4920.3	2.82852E-04	0.62829	-1.00012E+06	3.52516E-02	1483.9
237.23	683.92	5112.0	2.76626E-04	0.62218	-9.86784E+05	3.46240E-02	1458.3
239.85	680.79	5291.7	2.70399E-04	0.61607	-9.72977E+05	3.39963E-02	1433.9
242.46	677.65	5457.5	2.64173E-04	0.60996	-9.58730E+05	3.33686E-02	1410.7
245.08	674.51	5610.7	2.57947E-04	0.60389	-9.44076E+05	3.27410E-02	1388.3
247.69	671.37	5750.2	2.51720E-04	0.59792	-9.29049E+05	3.21133E-02	1366.7
250.31	668.23	5877.6	2.45494E-04	0.59194	-9.13683E+05	3.14856E-02	1345.6
252.93	665.09	5990.4	2.39267E-04	0.58596	-8.98014E+05	3.08580E-02	1325.1
255.54	661.91	6091.8	2.33041E-04	0.57999	-8.82074E+05	3.02303E-02	1313.5
258.16	658.25	6178.9	2.26814E-04	0.57401	-8.65900E+05	2.96026E-02	1343.1
260.77	654.59	6251.7	2.20588E-04	0.56804	-8.49526E+05	2.89750E-02	1322.5
263.39	650.93	6309.8	2.14361E-04	0.56206	-8.32992E+05	2.83473E-02	1302.4
266.00	647.26	6355.6	2.08135E-04	0.55609	-8.16330E+05	2.77196E-02	1282.7
268.62	643.60	6387.1	2.01908E-04	0.55011	-7.99577E+05	2.70920E-02	1263.3
271.23	639.94	6405.7	1.95682E-04	0.54413	-7.82769E+05	2.64643E-02	1244.2
273.85	636.28	6409.5	1.89455E-04	0.53816	-7.65940E+05	2.58366E-02	1225.3
276.46	632.61	6400.2	1.83229E-04	0.53218	-7.49127E+05	2.52090E-02	1206.6
279.08	628.95	6377.3	1.77909E-04	0.52613	-7.32365E+05	2.45813E-02	1188.2
281.69	625.29	6340.8	1.73239E-04	0.52002	-7.15688E+05	2.39536E-02	1170.0
284.31	621.62	6291.5	1.68569E-04	0.51391	-6.99131E+05	2.33260E-02	1151.9
286.92	617.96	6228.3	1.63899E-04	0.50779	-6.82730E+05	2.26983E-02	1133.9
289.54	614.30	6152.7	1.59229E-04	0.50168	-6.66516E+05	2.20706E-02	1116.1
292.15	610.64	6064.3	1.54559E-04	0.49557	-6.50521E+05	2.14430E-02	1098.5
294.77	606.97	5963.8	1.49890E-04	0.48946	-6.34779E+05	2.08153E-02	1081.0
297.38	603.31	5852.4	1.45220E-04	0.48335	-6.19319E+05	2.01876E-02	1063.6
300.00	599.65	5729.7	1.40697E-04	0.47724	-6.04169E+05	1.95600E-02	1046.3

MODEL = SAMPLE
PRPMAP

FLUINT SAMPLE MODEL

VAPOR TRANSPORT PROPERTIES USER DESCRIBED SIMPLIFIED TWO-PHASE 7717

TEMP.	VISCOSITY	CONDUCT.
195.39	7.56607E-06	1.14332E-02
198.00	7.66335E-06	1.18407E-02
200.62	7.76064E-06	1.22481E-02
203.23	7.85793E-06	1.26556E-02
205.85	7.95522E-06	1.30630E-02
208.47	8.05251E-06	1.34704E-02
211.08	8.14980E-06	1.38779E-02
213.70	8.24708E-06	1.42853E-02
216.31	8.34437E-06	1.46927E-02
218.93	8.44166E-06	1.51002E-02
221.54	8.53895E-06	1.55076E-02
224.16	8.63624E-06	1.59150E-02
226.77	8.73353E-06	1.63225E-02
229.39	8.83081E-06	1.67299E-02
232.00	8.92810E-06	1.71373E-02
234.62	9.02539E-06	1.75448E-02
237.23	9.12268E-06	1.79522E-02
239.85	9.21997E-06	1.83596E-02
242.46	9.31726E-06	1.87671E-02
245.08	9.41454E-06	1.91745E-02
247.69	9.51183E-06	1.95819E-02
250.31	9.60912E-06	1.99894E-02
252.93	9.70641E-06	2.03968E-02
255.54	9.80370E-06	2.08042E-02
258.16	9.90099E-06	2.12117E-02
260.77	9.99827E-06	2.16191E-02
263.39	1.00956E-05	2.20265E-02
266.00	1.01929E-05	2.24340E-02
268.62	1.02901E-05	2.28414E-02
271.23	1.03874E-05	2.32488E-02
273.85	1.04847E-05	2.36563E-02
276.46	1.05820E-05	2.40637E-02
279.08	1.06793E-05	2.44711E-02
281.69	1.07766E-05	2.48786E-02
284.31	1.08739E-05	2.52860E-02
286.92	1.09712E-05	2.56935E-02
289.54	1.10684E-05	2.61009E-02
292.15	1.11657E-05	2.65083E-02
294.77	1.12630E-05	2.69158E-02
297.38	1.13603E-05	2.73232E-02
300.00	1.14597E-05	2.77860E-02

MODEL = SAMPLE
PRPMAP

FLUINT SAMPLE MODEL

TABLE OF VAPOR SPECIFIC HEAT FOR USER DESCRIBED SIMPLIFIED TWO-PHASE 7717

TEMP.	PRESSURES						
	6043.9	1.82814E+05	3.59585E+05	5.36355E+05	7.13125E+05	8.89895E+05	1.06667E+06
195.39	1968.1						
198.00	1971.0						
200.62	1973.9						
203.23	1976.9						
205.85	1979.8						
208.47	1982.7						
211.08	1985.7						
213.70	1988.6						
216.31	1991.5						
218.93	1994.5						
221.54	1997.4						
224.16	2000.4						
226.77	2003.3						
229.39	2006.2						
232.00	2009.2						
234.62	2012.1						
237.23	2015.1						
239.85	2018.0						
242.46	2020.9						
245.08	2023.9						
247.69	2026.8						
250.31	2029.8						
252.93	2032.7	2049.9					
255.54	2035.7	2052.5					
258.16	2038.6	2055.1					
260.77	2041.6	2057.7					
263.39	2044.5	2060.3					
266.00	2047.5	2062.9					
268.62	2050.9	2066.1	2082.5				
271.23	2054.5	2069.4	2085.4				
273.85	2058.2	2072.7	2088.4				
276.46	2061.8	2076.0	2091.4				
279.08	2065.4	2079.4	2094.4	2110.5			
281.69	2069.0	2082.7	2097.4	2113.2			
284.31	2072.6	2086.1	2100.4	2115.9			
286.92	2076.2	2089.4	2103.5	2118.6	2134.8		
289.54	2079.8	2092.8	2106.6	2121.4	2137.2		
292.15	2083.4	2096.1	2109.7	2124.1	2139.6		
294.77	2087.0	2099.5	2112.8	2126.9	2142.0	2158.2	
297.38	2090.6	2102.9	2115.9	2129.8	2144.5	2160.3	
300.00	2094.3	2106.3	2119.1	2132.7	2147.1	2162.5	2179.0

MODEL = SAMPLE
PRPMAP

FLUINT SAMPLE MODEL

TABLE OF VAPOR SPECIFIC VOLUME FOR USER DESCRIBED SIMPLIFIED TWO-PHASE 7717

TEMP.	PRESSURES						
	6043.9	1.82814E+05	3.59585E+05	5.36355E+05	7.13125E+05	8.89895E+05	1.06667E+06
195.39	15.791						
198.00	16.003						
200.62	16.214						
203.23	16.426						
205.85	16.638						
208.47	16.850						
211.08	17.061						
213.70	17.273						
216.31	17.485						
218.93	17.696						
221.54	17.908						
224.16	18.120						
226.77	18.332						
229.39	18.543						
232.00	18.755						
234.62	18.967						
237.23	19.178						
239.85	19.390						
242.46	19.602						
245.08	19.813						
247.69	20.025						
250.31	20.237						
252.93	20.448	0.66661					
255.54	20.660	0.67373					
258.16	20.872	0.68085					
260.77	21.083	0.68796					
263.39	21.295	0.69507					
266.00	21.507	0.70218					
268.62	21.718	0.70929	0.35606				
271.23	21.930	0.71639	0.35973				
273.85	22.141	0.72349	0.36340				
276.46	22.353	0.73059	0.36707				
279.08	22.565	0.73769	0.37073	0.24559			
281.69	22.776	0.74479	0.37439	0.24808			
284.31	22.988	0.75188	0.37805	0.25057			
286.92	23.200	0.75897	0.38171	0.25306	0.18814		
289.54	23.411	0.76606	0.38536	0.25554	0.19004		
292.15	23.623	0.77315	0.38901	0.25803	0.19194		
294.77	23.835	0.78024	0.39267	0.26051	0.19383	0.15361	
297.38	24.046	0.78732	0.39631	0.26299	0.19572	0.15515	
300.00	24.258	0.79440	0.39996	0.26547	0.19761	0.15668	0.12929

MODEL = SAMPLE
PRPMAP

FLUINT SAMPLE MODEL

TABLE OF VAPOR ENTHALPY FOR USER DESCRIBED SIMPLIFIED TWO-PHASE 7717

TEMP.	PRESSURES						
	6043.9	1.82814E+05	3.59585E+05	5.36355E+05	7.13125E+05	8.89895E+05	1.06667E+06
195.39	3.62602E+05						
198.00	3.67753E+05						
200.62	3.72911E+05						
203.23	3.78078E+05						
205.85	3.83252E+05						
208.47	3.88433E+05						
211.08	3.93622E+05						
213.70	3.98819E+05						
216.31	4.04024E+05						
218.93	4.09236E+05						
221.54	4.14456E+05						
224.16	4.19684E+05						
226.77	4.24919E+05						
229.39	4.30162E+05						
232.00	4.35413E+05						
234.62	4.40671E+05						
237.23	4.45937E+05						
239.85	4.51211E+05						
242.46	4.56492E+05						
245.08	4.61782E+05						
247.69	4.67079E+05						
250.31	4.72383E+05						
252.93	4.77695E+05	4.73862E+05					
255.54	4.83015E+05	4.79227E+05					
258.16	4.88343E+05	4.84598E+05					
260.77	4.93678E+05	4.89976E+05					
263.39	4.99022E+05	4.95361E+05					
266.00	5.04372E+05	5.00752E+05					
268.62	5.09731E+05	5.06152E+05	5.02483E+05				
271.23	5.15100E+05	5.11559E+05	5.07933E+05				
273.85	5.20478E+05	5.16976E+05	5.13391E+05				
276.46	5.25865E+05	5.22401E+05	5.18857E+05				
279.08	5.31262E+05	5.27835E+05	5.24330E+05	5.20742E+05			
281.69	5.36668E+05	5.33277E+05	5.29811E+05	5.26265E+05			
284.31	5.42084E+05	5.38728E+05	5.35301E+05	5.31796E+05			
286.92	5.47509E+05	5.44188E+05	5.40798E+05	5.37333E+05	5.33788E+05		
289.54	5.52943E+05	5.49657E+05	5.46303E+05	5.42877E+05	5.39374E+05		
292.15	5.58387E+05	5.55135E+05	5.51817E+05	5.48428E+05	5.44966E+05		
294.77	5.63841E+05	5.60621E+05	5.57338E+05	5.53987E+05	5.50565E+05	5.47066E+05	
297.38	5.69303E+05	5.66116E+05	5.62868E+05	5.59553E+05	5.56170E+05	5.52713E+05	
300.00	5.74776E+05	5.71620E+05	5.68405E+05	5.65127E+05	5.61782E+05	5.58366E+05	5.54873E+05

MODEL = SAMPLE
PRPMAP

FLUINT SAMPLE MODEL

TABLE OF VAPOR ENTROPY FOR USER DESCRIBED SIMPLIFIED TWO-PHASE 7717

TEMP.	PRESSURES						
	6043.9	1.82814E+05	3.59585E+05	5.36355E+05	7.13125E+05	8.89895E+05	1.06667E+06
195.39	1348.9						
198.00	1375.1						
200.62	1401.0						
203.23	1426.6						
205.85	1451.9						
208.47	1476.9						
211.08	1501.6						
213.70	1526.1						
216.31	1550.3						
218.93	1574.2						
221.54	1597.9						
224.16	1621.4						
226.77	1644.6						
229.39	1667.6						
232.00	1690.4						
234.62	1712.9						
237.23	1735.2						
239.85	1757.3						
242.46	1779.2						
245.08	1800.9						
247.69	1822.4						
250.31	1843.7						
252.93	1864.9	189.72					
255.54	1885.8	210.82					
258.16	1906.5	231.73					
260.77	1927.1	252.46					
263.39	1947.5	273.01					
266.00	1967.7	293.38					
268.62	1987.7	313.58	-24.816				
271.23	2007.6	333.61	-4.6242				
273.85	2027.4	353.49	15.402				
276.46	2046.9	373.20	35.266				
279.08	2066.4	392.76	54.971	-147.82			
281.69	2085.6	412.17	74.520	-128.13			
284.31	2104.8	431.44	93.917	-108.59			
286.92	2123.8	450.55	113.16	-89.198	-235.57		
289.54	2142.6	469.53	132.26	-69.963	-216.19		
292.15	2161.4	488.36	151.22	-50.875	-196.96		
294.77	2179.9	507.06	170.04	-31.933	-177.88	-293.04	
297.38	2198.4	525.62	188.71	-13.133	-158.95	-273.97	
300.00	2216.7	544.04	207.25	5.5277	-140.16	-255.05	-350.45

MODEL = SAMPLE
PRPMAP

FLUINT SAMPLE MODEL

TABLE OF VAPOR SPEED OF SOUND FOR USER DESCRIBED SIMPLIFIED TWO-PHASE 7717

TEMP.	PRESSURES						
	6043.9	1.82814E+05	3.59585E+05	5.36355E+05	7.13125E+05	8.89895E+05	1.06667E+06
195.39	346.04						
198.00	348.30						
200.62	350.54						
203.23	352.78						
205.85	354.99						
208.47	357.20						
211.08	359.38						
213.70	361.56						
216.31	363.71						
218.93	365.86						
221.54	367.98						
224.16	370.10						
226.77	372.21						
229.39	374.30						
232.00	376.37						
234.62	378.44						
237.23	380.49						
239.85	382.53						
242.46	384.56						
245.08	386.58						
247.69	388.59						
250.31	390.57						
252.93	392.56	386.11					
255.54	394.53	388.16					
258.16	396.50	390.19					
260.77	398.45	392.21					
263.39	400.39	394.21					
266.00	402.31	396.20					
268.62	404.22	398.18	394.01				
271.23	406.12	400.14	396.04				
273.85	408.00	402.08	398.07				
276.46	409.87	404.02	400.07				
279.08	411.75	405.94	402.07	398.06			
281.69	413.59	407.85	404.05	400.12			
284.31	415.44	409.76	406.02	402.16			
286.92	417.30	411.65	407.99	404.19	400.28		
289.54	419.12	413.53	409.92	406.21	402.37		
292.15	420.95	415.40	411.86	408.20	404.44		
294.77	422.76	417.26	413.78	410.19	406.50	402.68	
297.38	424.56	419.11	415.69	412.17	408.53	404.79	
300.00	426.35	420.95	417.58	414.12	410.56	406.88	403.09

MODEL = SAMPLE
PRPMAP

FLUINT SAMPLE MODEL

PROPERTY MAPPING OF STANDARD TWO-PHASE LIBRARY FLUID 717

MINIMUM PRESSURE:	6045.6		
MAXIMUM GAS PRESSURE:	1.12801E+07	PRES. LIMIT FOR PRPMAP:	1.06083E+06
MINIMUM TEMPERATURE:	195.44		
MAXIMUM GAS TEMPERATURE:	1216.8	GAS TEMP. LIMIT FOR PRPMAP:	300.00
MAXIMUM LIQUID TEMPERATURE:	405.60	LIQ. TEMP. LIMIT FOR PRPMAP:	300.00
CRITICAL TEMPERATURE:	405.60		
CRITICAL PRESSURE:	1.12824E+07		
CRITICAL DENSITY:	235.57		

MODEL = SAMPLE
PRPMAP

FLUINT SAMPLE MODEL

SATURATION DOME DATA FOR STANDARD TWO-PHASE LIBRARY FLUID 717

TEMP.	SAT. PRESS.	DERIV. DPDT	HEAT OF VAP.	VAPOR SP. VOL.
195.44	6045.6	485.30	1.49143E+06	15.726
198.06	7433.4	578.73	1.48458E+06	12.953
200.67	9083.3	686.17	1.47771E+06	10.733
203.29	11034.	809.07	1.47084E+06	8.9441
205.90	13328.	948.96	1.46394E+06	7.4938
208.51	16011.	1107.4	1.45702E+06	6.3113
211.13	19135.	1286.1	1.45006E+06	5.3419
213.74	22753.	1486.6	1.44307E+06	4.5429
216.36	26927.	1710.8	1.43602E+06	3.8810
218.97	31719.	1960.5	1.42893E+06	3.3301
221.58	37199.	2237.3	1.42178E+06	2.8693
224.20	43441.	2543.3	1.41456E+06	2.4822
226.81	50522.	2880.3	1.40728E+06	2.1556
229.43	58527.	3250.2	1.39992E+06	1.8788
232.04	67544.	3654.9	1.39249E+06	1.6434
234.65	77666.	4096.2	1.38497E+06	1.4424
237.27	88992.	4576.2	1.37736E+06	1.2700
239.88	1.01624E+05	5096.6	1.36966E+06	1.1218
242.49	1.15673E+05	5659.5	1.36187E+06	0.99380
245.11	1.31250E+05	6266.7	1.35397E+06	0.88297
247.72	1.48474E+05	6920.0	1.34597E+06	0.78666
250.34	1.67468E+05	7621.3	1.33786E+06	0.70272
252.95	1.88359E+05	8372.4	1.32964E+06	0.62935
255.56	2.11281E+05	9175.1	1.32131E+06	0.56501
258.18	2.36371E+05	10031.	1.31286E+06	0.50845
260.79	2.63770E+05	10942.	1.30428E+06	0.45859
263.41	2.93623E+05	11910.	1.29558E+06	0.41452
266.02	3.26082E+05	12936.	1.28675E+06	0.37547
268.63	3.61301E+05	14022.	1.27779E+06	0.34078
271.25	3.99439E+05	15170.	1.26869E+06	0.30989
273.86	4.40659E+05	16380.	1.25945E+06	0.28233
276.48	4.85127E+05	17655.	1.25007E+06	0.25768
279.09	5.33014E+05	18996.	1.24054E+06	0.23558
281.70	5.84494E+05	20405.	1.23086E+06	0.21573
284.32	6.39744E+05	21882.	1.22102E+06	0.19787
286.93	6.98947E+05	23429.	1.21102E+06	0.18176
289.54	7.62286E+05	25047.	1.20086E+06	0.16721
292.16	8.29951E+05	26738.	1.19052E+06	0.15403
294.77	9.02131E+05	28503.	1.18001E+06	0.14209
297.39	9.79024E+05	30343.	1.16931E+06	0.13124
300.00	1.06083E+06	32260.	1.15842E+06	0.12136

MODEL = SAMPLE FLUINT SAMPLE MODEL
PRPMAP

SATURATED LIQUID PROPERTIES FOR STANDARD TWO-PHASE LIBRARY FLUID 717

TEMP.	DENSITY	SP. HEAT	VISCOSITY	CONDUCT.	ENTHALPY	SURFACE TENS.	SOUND SPEED
195.44	734.14	4510.8	3.79444E-04	0.71966	-1.68936E+05	4.65508E-02	1964.7
198.06	731.17	4502.6	3.75918E-04	0.71360	-1.57153E+05	4.57976E-02	1953.4
200.67	728.19	4495.5	3.71721E-04	0.70753	-1.45393E+05	4.50501E-02	1941.9
203.29	725.19	4488.6	3.66933E-04	0.70147	-1.33653E+05	4.43081E-02	1930.3
205.90	722.18	4482.1	3.61628E-04	0.69541	-1.21930E+05	4.35715E-02	1918.6
208.51	719.15	4478.5	3.55879E-04	0.68934	-1.10220E+05	4.28403E-02	1906.5
211.13	716.12	4474.4	3.49751E-04	0.68328	-98519.	4.21142E-02	1894.4
213.74	713.06	4472.3	3.43306E-04	0.67721	-86826.	4.13932E-02	1881.8
216.36	710.00	4469.5	3.36600E-04	0.67115	-75136.	4.06770E-02	1869.2
218.97	706.92	4468.9	3.29686E-04	0.66508	-63447.	3.99657E-02	1856.3
221.58	703.82	4469.2	3.22609E-04	0.65902	-51758.	3.92592E-02	1843.2
224.20	700.71	4470.3	3.15413E-04	0.65296	-40065.	3.85571E-02	1829.9
226.81	697.58	4472.1	3.08136E-04	0.64689	-28367.	3.78595E-02	1816.4
229.43	694.44	4475.4	3.00814E-04	0.64083	-16662.	3.71662E-02	1802.6
232.04	691.27	4477.2	2.93475E-04	0.63476	-4948.2	3.64771E-02	1788.8
234.65	688.09	4480.8	2.86147E-04	0.62870	6775.8	3.57921E-02	1774.7
237.27	684.89	4484.4	2.78855E-04	0.62263	18511.	3.51109E-02	1760.5
239.88	681.67	4488.6	2.71620E-04	0.61657	30259.	3.44337E-02	1746.1
242.49	678.43	4493.6	2.64459E-04	0.61051	42021.	3.37601E-02	1731.5
245.11	675.17	4498.5	2.57388E-04	0.60444	53798.	3.30901E-02	1716.7
247.72	671.89	4504.2	2.50421E-04	0.59838	65591.	3.24236E-02	1701.7
250.34	668.59	4509.6	2.43569E-04	0.59231	77401.	3.17604E-02	1686.7
252.95	665.26	4515.2	2.36841E-04	0.58625	89228.	3.11005E-02	1671.4
255.56	661.91	4520.7	2.30246E-04	0.58018	1.01073E+05	3.04437E-02	1656.1
258.18	658.54	4527.2	2.23790E-04	0.57412	1.12938E+05	2.97900E-02	1640.7
260.79	655.13	4533.4	2.17477E-04	0.56806	1.24824E+05	2.91392E-02	1625.0
263.41	651.71	4540.4	2.11311E-04	0.56199	1.36730E+05	2.84914E-02	1609.1
266.02	648.25	4546.8	2.05296E-04	0.55593	1.48658E+05	2.78463E-02	1593.1
268.63	644.76	4554.3	1.99433E-04	0.54986	1.60611E+05	2.72039E-02	1576.9
271.25	641.25	4563.2	1.93723E-04	0.54380	1.72588E+05	2.65642E-02	1560.5
273.86	637.70	4571.1	1.88166E-04	0.53773	1.84590E+05	2.59271E-02	1544.1
276.48	634.13	4578.4	1.82762E-04	0.53167	1.96620E+05	2.52925E-02	1527.4
279.09	630.52	4588.5	1.77510E-04	0.52561	2.08680E+05	2.46603E-02	1510.4
281.70	626.87	4598.7	1.72408E-04	0.51954	2.20771E+05	2.40305E-02	1493.3
284.32	623.19	4607.8	1.67455E-04	0.51348	2.32893E+05	2.34032E-02	1476.2
286.93	619.47	4619.3	1.62649E-04	0.50741	2.45051E+05	2.27783E-02	1458.7
289.54	615.70	4631.1	1.57986E-04	0.50135	2.57247E+05	2.21556E-02	1441.3
292.16	611.90	4643.4	1.53466E-04	0.49528	2.69482E+05	2.15353E-02	1423.6
294.77	608.06	4657.3	1.49083E-04	0.48922	2.81760E+05	2.09174E-02	1405.6
297.39	604.17	4672.9	1.44837E-04	0.48316	2.94084E+05	2.03018E-02	1387.1
300.00	600.23	4688.1	1.40722E-04	0.47709	3.06457E+05	1.96886E-02	1368.8

MODEL = SAMPLE FLUINT SAMPLE MODEL
PRPMAP

VAPOR TRANSPORT PROPERTIES STANDARD TWO-PHASE LIBRARY FLUID 717

TEMP.	VISCOSITY	CONDUCT.
195.44	7.13747E-06	1.50866E-02
198.06	7.18029E-06	1.52907E-02
200.67	7.22654E-06	1.54966E-02
203.29	7.27599E-06	1.57043E-02
205.90	7.32845E-06	1.59138E-02
208.51	7.38375E-06	1.61250E-02
211.13	7.44171E-06	1.63381E-02
213.74	7.50217E-06	1.65529E-02
216.36	7.56499E-06	1.67695E-02
218.97	7.63002E-06	1.69879E-02
221.58	7.69713E-06	1.72081E-02
224.20	7.76620E-06	1.74301E-02
226.81	7.83710E-06	1.76539E-02
229.43	7.90974E-06	1.78795E-02
232.04	7.98400E-06	1.81068E-02
234.65	8.05978E-06	1.83359E-02
237.27	8.13701E-06	1.85668E-02
239.88	8.21559E-06	1.87995E-02
242.49	8.29543E-06	1.90340E-02
245.11	8.37646E-06	1.92703E-02
247.72	8.45862E-06	1.95084E-02
250.34	8.54182E-06	1.97482E-02
252.95	8.62600E-06	1.99898E-02
255.56	8.71111E-06	2.02333E-02
258.18	8.79709E-06	2.04785E-02
260.79	8.88387E-06	2.07255E-02
263.41	8.97142E-06	2.09742E-02
266.02	9.05967E-06	2.12248E-02
268.63	9.14858E-06	2.14772E-02
271.25	9.23811E-06	2.17313E-02
273.86	9.32822E-06	2.19872E-02
276.48	9.41887E-06	2.22449E-02
279.09	9.51001E-06	2.25044E-02
281.70	9.60162E-06	2.27657E-02
284.32	9.69366E-06	2.30288E-02
286.93	9.78610E-06	2.32936E-02
289.54	9.87890E-06	2.35603E-02
292.16	9.97204E-06	2.38287E-02
294.77	1.00655E-05	2.40989E-02
297.39	1.01592E-05	2.43709E-02
300.00	1.02532E-05	2.46447E-02

MODEL = SAMPLE FLUINT SAMPLE MODEL
PRPMAP

TABLE OF VAPOR SPECIFIC HEAT FOR STANDARD TWO-PHASE LIBRARY FLUID 717

TEMP.	PRESSURES						
	6045.6	1.81842E+05	3.57639E+05	5.33436E+05	7.09233E+05	8.85030E+05	1.06083E+06
195.44	1998.7						
198.06	2000.0						
200.67	2001.4						
203.29	2002.5						
205.90	2003.8						
208.51	2005.0						
211.13	2006.5						
213.74	2008.6						
216.36	2010.4						
218.97	2012.2						
221.58	2014.0						
224.20	2015.8						
226.81	2018.0						
229.43	2020.0						
232.04	2022.5						
234.65	2025.0						
237.27	2027.0						
239.88	2029.5						
242.49	2032.2						
245.11	2034.7						
247.72	2037.1						
250.34	2040.1						
252.95	2042.8	2287.8					
255.56	2045.7	2280.4					
258.18	2048.6	2273.4					
260.79	2051.3	2266.6					
263.41	2054.7	2261.2					
266.02	2057.8	2255.6					
268.63	2061.0	2251.1	2469.6				
271.25	2063.9	2246.6	2455.4				
273.86	2067.3	2242.8	2441.7				
276.48	2070.9	2238.8	2428.9				
279.09	2074.0	2235.8	2417.2				
281.70	2077.6	2232.9	2406.6	2603.0			
284.32	2081.0	2230.4	2396.2	2583.2			
286.93	2084.6	2228.2	2387.2	2564.5			
289.54	2088.2	2226.4	2378.5	2547.9	2738.5		
292.16	2091.8	2225.0	2370.4	2531.7	2712.4		
294.77	2095.9	2223.7	2363.2	2517.1	2688.3	2882.2	
297.39	2099.2	2222.3	2356.4	2503.3	2666.2	2847.8	
300.00	2103.3	2221.4	2349.7	2490.1	2644.9	2817.2	3011.6

MODEL = SAMPLE FLUINT SAMPLE MODEL
PRPMAP

TABLE OF VAPOR SPECIFIC VOLUME FOR STANDARD TWO-PHASE LIBRARY FLUID 717

TEMP.	PRESSURES						
	6045.6	1.81842E+05	3.57639E+05	5.33436E+05	7.09233E+05	8.85030E+05	1.06083E+06
195.44	15.726						
198.06	15.939						
200.67	16.152						
203.29	16.365						
205.90	16.578						
208.51	16.791						
211.13	17.004						
213.74	17.216						
216.36	17.429						
218.97	17.642						
221.58	17.854						
224.20	18.066						
226.81	18.279						
229.43	18.491						
232.04	18.703						
234.65	18.916						
237.27	19.128						
239.88	19.340						
242.49	19.552						
245.11	19.764						
247.72	19.976						
250.34	20.188						
252.95	20.400	0.65288					
255.56	20.611	0.66075					
258.18	20.823	0.66859					
260.79	21.035	0.67640					
263.41	21.247	0.68417					
266.02	21.459	0.69190					
268.63	21.670	0.69961	0.34451				
271.25	21.882	0.70728	0.34678				
273.86	22.094	0.71493	0.35302				
276.48	22.305	0.72255	0.35723				
279.09	22.517	0.73014	0.36142				
281.70	22.728	0.73772	0.36557	0.23839			
284.32	22.940	0.74526	0.36970	0.24138			
286.93	23.152	0.75279	0.37381	0.24434			
289.54	23.363	0.76030	0.37789	0.24728	0.18121		
292.16	23.575	0.76778	0.38195	0.25019	0.18356		
294.77	23.786	0.77525	0.38599	0.25308	0.18588	0.14521	
297.39	23.998	0.78270	0.39001	0.25595	0.18818	0.14718	
300.00	24.209	0.79014	0.39401	0.25880	0.19046	0.14913	0.12136

MODEL = SAMPLE FLUINT SAMPLE MODEL
PRPMAP

TABLE OF VAPOR ENTHALPY FOR STANDARD TWO-PHASE LIBRARY FLUID 717

TEMP.	PRESSURES						
	6045.6	1.81842E+05	3.57639E+05	5.33436E+05	7.09233E+05	8.85030E+05	1.06083E+06
195.44	1.32250E+06						
198.06	1.32772E+06						
200.67	1.33295E+06						
203.29	1.33818E+06						
205.90	1.34342E+06						
208.51	1.34866E+06						
211.13	1.35390E+06						
213.74	1.35915E+06						
216.36	1.36440E+06						
218.97	1.36966E+06						
221.58	1.37492E+06						
224.20	1.38018E+06						
226.81	1.38545E+06						
229.43	1.39073E+06						
232.04	1.39601E+06						
234.65	1.40130E+06						
237.27	1.40660E+06						
239.88	1.41190E+06						
242.49	1.41721E+06						
245.11	1.42252E+06						
247.72	1.42784E+06						
250.34	1.43317E+06						
252.95	1.43850E+06	1.41960E+06					
255.56	1.44385E+06	1.42557E+06					
258.18	1.44920E+06	1.43153E+06					
260.79	1.45456E+06	1.43746E+06					
263.41	1.45992E+06	1.44338E+06					
266.02	1.46529E+06	1.44929E+06					
268.63	1.47068E+06	1.45518E+06	1.43875E+06				
271.25	1.47607E+06	1.46106E+06	1.44519E+06				
273.86	1.48147E+06	1.46693E+06	1.45160E+06				
276.48	1.48687E+06	1.47279E+06	1.45797E+06				
279.09	1.49229E+06	1.47864E+06	1.46430E+06				
281.70	1.49771E+06	1.48448E+06	1.47061E+06	1.45602E+06			
284.32	1.50315E+06	1.49031E+06	1.47689E+06	1.46281E+06			
286.93	1.50859E+06	1.49614E+06	1.48315E+06	1.46954E+06			
289.54	1.51404E+06	1.50196E+06	1.48938E+06	1.47623E+06	1.46242E+06		
292.16	1.51951E+06	1.50778E+06	1.49559E+06	1.48287E+06	1.46955E+06		
294.77	1.52498E+06	1.51359E+06	1.50178E+06	1.48947E+06	1.47661E+06	1.46312E+06	
297.39	1.53046E+06	1.51940E+06	1.50795E+06	1.49604E+06	1.48362E+06	1.47062E+06	
300.00	1.53595E+06	1.52521E+06	1.51410E+06	1.50257E+06	1.49056E+06	1.47803E+06	1.46488E+06

MODEL = SAMPLE FLUINT SAMPLE MODEL
PRPMAP

TABLE OF VAPOR ENTROPY FOR STANDARD TWO-PHASE LIBRARY FLUID 717

TEMP.	PRESSURES						
	6045.6	1.81842E+05	3.57639E+05	5.33436E+05	7.09233E+05	8.85030E+05	1.06083E+06
195.44	6840.9						
198.06	6867.4						
200.67	6893.7						
203.29	6919.6						
205.90	6945.2						
208.51	6970.4						
211.13	6995.4						
213.74	7020.1						
216.36	7044.6						
218.97	7068.7						
221.58	7092.6						
224.20	7116.2						
226.81	7139.6						
229.43	7162.7						
232.04	7185.6						
234.65	7208.3						
237.27	7230.7						
239.88	7252.9						
242.49	7274.9						
245.11	7296.7						
247.72	7318.3						
250.34	7339.7						
252.95	7360.9	5642.5					
255.56	7381.9	5666.0					
258.18	7402.8	5689.1					
260.79	7423.4	5712.0					
263.41	7443.9	5734.6					
266.02	7464.2	5756.9					
268.63	7484.3	5778.9	5401.9				
271.25	7504.3	5800.7	5425.8				
273.86	7524.1	5822.3	5449.3				
276.48	7543.8	5843.5	5472.4				
279.09	7563.3	5864.6	5495.2				
281.70	7582.6	5885.4	5517.7	5282.7			
284.32	7601.8	5906.1	5539.9	5306.7			
286.93	7620.9	5926.5	5561.8	5330.3			
289.54	7639.8	5946.7	5583.4	5353.5	5177.6		
292.16	7658.6	5966.7	5604.8	5376.3	5202.1		
294.77	7677.2	5986.5	5625.9	5398.8	5226.2	5082.5	
297.39	7695.7	6006.1	5646.7	5421.0	5249.8	5107.9	
300.00	7714.1	6025.5	5667.3	5442.8	5273.1	5132.7	5010.1

TABLE OF VAPOR SPEED OF SOUND FOR STANDARD TWO-PHASE LIBRARY FLUID 7717

TEMP.	PRESSURES						
	6045.6	1.81842E+05	3.57639E+05	5.33436E+05	7.09233E+05	8.85030E+05	1.06083E+06
195.44	344.35						
198.06	346.31						
200.67	348.62						
203.29	351.11						
205.90	353.21						
208.51	355.35						
211.13	357.60						
213.74	359.96						
216.36	362.05						
218.97	364.26						
221.58	366.42						
224.20	368.63						
226.81	370.76						
229.43	372.86						
232.04	374.81						
234.65	376.93						
237.27	379.14						
239.88	381.19						
242.49	383.08						
245.11	385.33						
247.72	387.16						
250.34	388.99						
252.95	391.17	380.25					
255.56	393.03	382.66					
258.18	394.97	384.90					
260.79	397.12	387.31					
263.41	398.94	389.64					
266.02	400.99	391.72					
268.63	402.92	394.02	384.88				
271.25	404.64	396.17	387.17				
273.86	406.41	398.36	389.74				
276.48	408.44	400.58	392.22				
279.09	410.50	402.64	394.59				
281.70	412.18	404.65	396.98	388.71			
284.32	414.04	406.69	399.50	391.61			
286.93	415.89	408.84	401.70	394.03			
289.54	417.71	410.77	403.98	396.49	388.57		
292.16	419.45	412.95	406.17	399.02	391.57		
294.77	421.21	414.72	408.56	401.69	394.32	386.39	
297.39	422.98	416.86	410.63	403.99	396.96	389.34	
300.00	424.97	418.94	412.88	406.40	399.61	392.34	384.58

FLUID SUBMODEL NAME = LOOP ; FLUID NO. = 7717

LOOPCT = 30
CONVERGENCE STATUS = SUBMODEL CONVERGED AS OF 28 ITERATIONS

LUMP PARAMETER TABULATION FOR SUBMODEL LOOP

LUMP	TYPE	TEMP	PRESSURE	QUALITY	DENSITY	ENTHALPY	HEAT RATE	MASS RATE	ENERGY RATE
1	TANK	300.0	1.0669E+06	0.5660	13.53	5.1825E+04	400.0	0.	1.4343E-02
2	TANK	300.0	1.0667E+06	0.7658	10.06	2.8346E+05	0.	0.	7.6294E-06
3	TANK	271.8	1.0667E+06	0.	639.1	-7.7815E+05	0.	0.	3.6163E-02
10	JUNC	300.0	1.0667E+06	0.4219	18.01	-1.1514E+05	-147.2	0.	8.1375E-02
20	JUNC	300.0	1.0667E+06	0.1112	63.07	-4.7531E+05	-133.1	0.	2.5208E-02
30	JUNC	288.6	1.0667E+06	0.	615.7	-6.7198E+05	-72.68	0.	-1.1757E-02
40	JUNC	271.6	1.0667E+06	0.	639.5	-7.7965E+05	-39.79	0.	-3.1357E-03
50	JUNC	262.2	1.0667E+06	0.	652.6	-8.3949E+05	-22.11	0.	1.4744E-03
200	JUNC	276.5	1.0669E+06	0.	632.6	-7.4815E+05	15.00	0.	0.
100	PLEN	271.7	1.0667E+06	0.	639.2	-7.7849E+05	0.	0.	0.

FLUINT SAMPLE MODEL

MODEL = SAMPLE
FASTIC

FLUID SUBMODEL NAME = LOOP ; FLUID NO. = 7717

LOOPCT = 30
CONVERGENCE STATUS = SUBMODEL CONVERGED AS OF 28 ITERATIONS

TIE PARAMETER TABULATION FOR SUBMODEL LOOP

TIE	TYPE	UA	QTIE	LUMP	TEMP.	NODE	TEMP.	PATH 1	FRACT	PATH 2	FRACT
10	HTNC	45.00	-147.2	10	300.0	RAD.10	296.7	10	1.0000	20	0.5000
20	HTNC	19.37	-133.1	20	300.0	RAD.20	293.1	20	0.5000	30	0.5000
30	HTNC	5.794	-72.68	30	288.6	RAD.30	276.0	30	0.5000	40	0.5000
40	HTNC	6.248	-39.79	40	271.6	RAD.40	265.2	40	0.5000	50	0.5000
50	HTNC	6.495	-22.11	50	262.2	RAD.50	258.8	50	0.5000	60	1.0000

FLUINT SAMPLE MODEL

MODEL = SAMPLE
FASTIC

FLUID SUBMODEL NAME = LOOP ; FLUID NO. = 7717

LOOPCT = 30
CONVERGENCE STATUS = SUBMODEL CONVERGED AS OF 28 ITERATIONS

PATH PARAMETER TABULATION FOR SUBMODEL LOOP

PATH	TYPE	LMP 1	LMP 2	DUP I	DUP J	STAT	XL UPSTRM	FLOWRATE	DELTA PRES	FLOW REGIMES OR REYNOLDS NO.	
1	TUBE	200	1	1.0	1.0	NORM	0.	5.0000E-04	67.73	434.39	UNKNOWN
2	TUBE	1	2	1.0	1.0	NORM	0.5660	5.0000E-04	138.9	UNKNOWN	UNKNOWN
3	CAPIL	2	3	1.0	1.0	DLS	0.	1.3048E-04	59.05		
10	STUBE	2	10	1.0	1.0	NORM	0.7658	3.6952E-04	20.96	UNKNOWN	UNKNOWN
20	STUBE	10	20	1.0	1.0	NORM	0.4219	3.6952E-04	26.71	UNKNOWN	UNKNOWN
30	STUBE	20	30	1.0	1.0	NORM	0.1112	3.6952E-04	8.585	UNKNOWN	365.39
40	STUBE	30	40	1.0	1.0	NORM	0.	3.6952E-04	1.038	365.39	301.75
50	STUBE	40	50	1.0	1.0	NORM	0.	3.6952E-04	2.170	301.75	270.71
60	STUBE	50	3	1.0	1.0	NORM	0.	3.6952E-04	0.5855	270.71	302.62
200	MFRSET	3	200	1.0	1.0	NORM	0.	5.0000E-04	-265.7		
100	STUBE	3	100	1.0	1.0	NORM	0.	0.	2.3283E-09	0.	0.

SUBMODEL NAME = RAD

ARLXCA =	1.000000E-02	ARLXCC =	1.01000	ATMPCA =	1.000000E+30	ATMPCC =	0.	BACKUP =	0.		
CSGFAC =	0.	CSGMAX =	0.	CSGMIN =	0.	DRLXCA =	1.000000E-02	DRLXCC =	1.01000		
DTIMEB =	1.000000E+30	DTIMEI =	0.	DTIMEL =	0.	DTIMEU =	0.	DTMPCA =	1.000000E+30		
DTMPCC =	0.	EBALNA =	0.	EBALNC =	-7.324219E-04	EBALSA =	1.000000E-02	EBALSC =	1.000000E+30		
ESUMIS =	0.	ESUMOS =	0.	EXTLIM =	50.0000	ITHOLD =	10	NARLXN =	0		
NATMPN =	0	NCGMAN =	0	NCSGMN =	0	NDRLXN =	0	NDTMPN =	0		
NEBALN =	0	NLOOPT =	0	ITEROT =	0	ITERXT =	3	OPEITR =	0		
OUTPUT =	1000.00										

GLOBAL CONTROL CONSTANTS

ABSZRO =	0.	SIGMA =	5.670000E-08	TIMEM =	0.	TIMEN =	0.	TIMEND =	0.
TIMEO =	0.								

MODEL = SAMPLE
FORWRD

FLUINT SAMPLE MODEL

FLUID SUBMODEL NAME = LOOP ; FLUID NO. = 7717

MAX TIME STEP = 4.73901 ; LIMITING TANK = 3 REASON = TEMP./DENSITY CHANGE LIMIT
LAST TIME STEP = 2.50000 VS. DTMAXF/DTMINF = 1.000000E+30 / 0. ; AVERAGE TIME STEP = 4.09390
PROBLEM TIME TIMEN = 60.0000 VS. TIMEND = 60.0000

LUMP PARAMETER TABULATION FOR SUBMODEL LOOP

LUMP	TYPE	TEMP	PRESSURE	QUALITY	DENSITY	ENTHALPY	HEAT RATE	MASS RATE	ENERGY RATE
1	TANK	300.0	1.0668E+06	0.5557	13.78	3.9987E+04	400.0	1.4169E-06	-0.7514
2	TANK	300.0	1.0667E+06	0.6634	11.58	1.6472E+05	0.	4.1929E-06	-2.582
3	TANK	269.7	1.0667E+06	0.	642.1	-7.9163E+05	0.	1.0393E-05	-36.26
10	JUNC	300.0	1.0667E+06	0.1823	40.10	-3.9284E+05	-232.9	0.	1.5259E-05
20	JUNC	299.4	1.0667E+06	0.	600.5	-6.0747E+05	-89.66	0.	0.
30	JUNC	276.3	1.0667E+06	0.	632.8	-7.4910E+05	-59.17	2.9104E-11	-3.0518E-05
40	JUNC	262.2	1.0667E+06	0.	652.6	-8.3922E+05	-37.65	0.	0.
50	JUNC	253.5	1.0667E+06	0.	664.4	-8.9315E+05	-22.53	0.	0.
200	JUNC	274.4	1.0669E+06	0.	635.5	-7.6163E+05	15.00	0.	0.
100	PLEN	269.8	1.0667E+06	0.	641.9	-7.9068E+05	0.	-1.6003E-05	12.65

MODEL = SAMPLE
FORWRD

FLUINT SAMPLE MODEL

FLUID SUBMODEL NAME = LOOP ; FLUID NO. = 7717

MAX TIME STEP = 4.73901 ; LIMITING TANK = 3 REASON = TEMP./DENSITY CHANGE LIMIT
LAST TIME STEP = 2.50000 VS. DTMAXF/DTMINF = 1.000000E+30 / 0. ; AVERAGE TIME STEP = 4.09390
PROBLEM TIME TIMEN = 60.0000 VS. TIMEND = 60.0000

TIE PARAMETER TABULATION FOR SUBMODEL LOOP

TIE	TYPE	UA	QTIE	LUMP	TEMP.	NODE	TEMP.	PATH 1	FRACT	PATH 2	FRACT
10	HTNC	27.64	-232.9	10	300.0	RAD.10	291.5	10	1.0000	20	0.5000
20	HTNC	5.498	-89.66	20	299.4	RAD.20	282.9	20	0.5000	30	0.5000
30	HTNC	6.121	-59.17	30	276.3	RAD.30	266.3	30	0.5000	40	0.5000
40	HTNC	6.494	-37.65	40	262.2	RAD.40	256.0	40	0.5000	50	0.5000
50	HTNC	6.725	-22.53	50	253.5	RAD.50	249.7	50	0.5000	60	1.0000

MODEL = SAMPLE FLUINT SAMPLE MODEL
FORWRD

FLUID SUBMODEL NAME = LOOP ; FLUID NO. = 7717

MAX TIME STEP = 4.73901 ; LIMITING TANK = 3 REASON = TEMP./DENSITY CHANGE LIMIT
LAST TIME STEP = 2.50000 VS. DTMAXF/DTMINF = 1.000000E+30 / 0. ; AVERAGE TIME STEP = 4.09390
PROBLEM TIME TIMEN = 60.0000 VS. TIMEND = 60.0000

PATH PARAMETER TABULATION FOR SUBMODEL LOOP

PATH	TYPE	LMP 1	LMP 2	DUP I	DUP J	STAT	XL UPSTRM	FLOWRATE	DELTA PRES	FLOW REGIMES OR REYNOLDS NO.	
1	TUBE	200	1	1.0	1.0	NORM	0.	5.0000E-04	67.11	422.82	UNKNOWN
2	TUBE	1	2	1.0	1.0	NORM	0.5557	4.9858E-04	133.8	UNKNOWN	UNKNOWN
3	CAPIL	2	3	1.0	1.0	DLS	0.	7.6629E-05	34.58		
10	STUBE	2	10	1.0	1.0	NORM	0.6634	4.1776E-04	16.97	UNKNOWN	UNKNOWN
20	STUBE	10	20	1.0	1.0	NORM	0.1823	4.1776E-04	12.73	UNKNOWN	469.60
30	STUBE	20	30	1.0	1.0	NORM	0.	4.1776E-04	1.468	469.60	362.24
40	STUBE	30	40	1.0	1.0	NORM	0.	4.1776E-04	1.286	362.24	306.19
50	STUBE	40	50	1.0	1.0	NORM	0.	4.1776E-04	1.429	306.19	279.52
60	STUBE	50	3	1.0	1.0	NORM	0.	4.1776E-04	0.6927	279.52	333.53
200	MFRSET	3	200	1.0	1.0	NORM	0.	5.0000E-04	-235.5		
100	STUBE	3	100	1.0	1.0	NORM	0.	-1.6003E-05	-0.2495	34.070	34.130

SUBMODEL NAME = RAD

ARLXCA =	1.000000E-02	ARLXCC =	0.	ATMPCA =	1.000000E+30	ATMPCC =	0.	BACKUP =	0.		
CSGFAC =	1.00000	CSGMAX =	96.8724	CSGMIN =	28.4817	DRLXCA =	1.000000E-02	DRLXCC =	0.		
DTIMEB =	1.000000E+30	DTIMEI =	0.	DTIMEL =	0.	DTIMEU =	0.	DTMPCA =	1.000000E+30		
DTMPCC =	0.	EBALNA =	0.	EBALNC =	1.251221E-03	EBALSA =	1.000000E-02	EBALSC =	0.		
ESUMIS =	414.936	ESUMOS =	414.936	EXTLIM =	50.0000	ITHOLD =	10	NARLXN =	0		
NATMPN =	0	NCGMAN =	20	NCSGMN =	10	NDRLXN =	0	NDTMPN =	0		
NEBALN =	0	NLOOPT =	100	ITEROT =	0	ITERXT =	3	OPEITR =	0		
OUTPUT =	1000.00										

GLOBAL CONTROL CONSTANTS

ABSZRO =	0.	SIGMA =	5.670000E-08	TIMEM =	58.7500	TIMEN =	60.0000	TIMEND =	120.000
TIMEO =	60.0000								

MODEL = SAMPLE FLUINT SAMPLE MODEL
FORWRD

FLUID SUBMODEL NAME = LOOP ; FLUID NO. = 7717

MAX TIME STEP (GROWTH LIMITED) = 5.34617 ; LAST CAUSE: TANK 2; FR CHANGE LIMIT IN PATH 10 (WAS 2.673)
LAST TIME STEP = 1.80643 VS. DTMAXF/DTMINF = 1.000000E+30 / 0. ; AVERAGE TIME STEP = 2.82799
PROBLEM TIME TIMEN = 120.000 VS. TIMEND = 120.000

LUMP PARAMETER TABULATION FOR SUBMODEL LOOP

LUMP	TYPE	TEMP	PRESSURE	QUALITY	DENSITY	ENTHALPY	HEAT RATE	MASS RATE	ENERGY RATE
1	TANK	300.0	1.0669E+06	0.5542	13.82	3.8206E+04	400.0	-1.2250E-06	0.6782
2	TANK	300.0	1.0667E+06	0.8443	9.139	3.7447E+05	0.	-9.1890E-06	5.759
3	TANK	269.9	1.0667E+06	0.	641.8	-7.9034E+05	0.	-9.2944E-06	29.33
10	JUNC	300.0	1.0667E+06	0.8191	9.416	3.4525E+05	-9.605	2.9104E-11	7.6294E-06
20	JUNC	300.0	1.0667E+06	0.3983	19.05	-1.4253E+05	-160.4	0.	0.
30	JUNC	300.0	1.0667E+06	1.0876E-02	327.3	-5.9156E+05	-147.6	-2.9104E-11	1.5259E-05
40	JUNC	275.8	1.0667E+06	0.	633.6	-7.5280E+05	-53.01	0.	0.
50	JUNC	264.3	1.0667E+06	0.	649.7	-8.2634E+05	-24.18	0.	-1.3351E-05
200	JUNC	274.6	1.0670E+06	0.	635.3	-7.6034E+05	15.00	0.	0.
100	PLEN	269.8	1.0667E+06	0.	641.9	-7.9096E+05	0.	1.9708E-05	-15.58

FLUID SUBMODEL NAME = LOOP ; FLUID NO. = 7717

MAX TIME STEP (GROWTH LIMITED) = 5.34617 ; LAST CAUSE: TANK 2; FR CHANGE LIMIT IN PATH 10 (WAS 2.673)
LAST TIME STEP = 1.80643 VS. DTMAXF/DTMINF = 1.000000E+30 / 0. ; AVERAGE TIME STEP = 2.82799
PROBLEM TIME TIMEN = 120.000 VS. TIMEND = 120.000

TIE PARAMETER TABULATION FOR SUBMODEL LOOP

TIE	TYPE	UA	QTIE	LUMP	TEMP.	NODE	TEMP.	PATH 1	FRACT	PATH 2	FRACT
10	HTNC	90.71	-9.605	10	300.0	RAD.10	299.9	10	1.0000	20	0.5000
20	HTNC	39.67	-160.4	20	300.0	RAD.20	296.3	20	0.5000	30	0.5000
30	HTNC	6.738	-147.6	30	300.0	RAD.30	278.6	30	0.5000	40	0.5000
40	HTNC	6.134	-53.01	40	275.8	RAD.40	267.6	40	0.5000	50	0.5000
50	HTNC	6.432	-24.18	50	264.3	RAD.50	261.1	50	0.5000	60	1.0000

FLUID SUBMODEL NAME = LOOP ; FLUID NO. = 7717

MAX TIME STEP (GROWTH LIMITED) = 5.34617 ; LAST CAUSE: TANK 2; FR CHANGE LIMIT IN PATH 10 (WAS 2.673)
LAST TIME STEP = 1.80643 VS. DTMAXF/DTMINF = 1.000000E+30 / 0. ; AVERAGE TIME STEP = 2.82799
PROBLEM TIME TIMEN = 120.000 VS. TIMEND = 120.000

PATH PARAMETER TABULATION FOR SUBMODEL LOOP

PATH	TYPE	LMP 1	LMP 2	DUP I	DUP J	STAT	XL UPSTRM	FLOWRATE	DELTA PRES	FLOW REGIMES OR REYNOLDS NO.	
1	TUBE	200	1	1.0	1.0	NORM	0.	5.0000E-04	66.92	423.89	UNKNOWN
2	TUBE	1	2	1.0	1.0	NORM	0.5542	5.0123E-04	140.0	UNKNOWN	UNKNOWN
3	CAPIL	2	3	1.0	1.0	DLS	0.	1.8163E-04	81.95		
10	STUBE	2	10	1.0	1.0	NORM	0.8443	3.2879E-04	23.80	UNKNOWN	UNKNOWN
20	STUBE	10	20	1.0	1.0	NORM	0.8191	3.2879E-04	38.48	UNKNOWN	UNKNOWN
30	STUBE	20	30	1.0	1.0	NORM	0.3983	3.2879E-04	16.28	UNKNOWN	UNKNOWN
40	STUBE	30	40	1.0	1.0	NORM	1.0876E-02	3.2879E-04	1.861	UNKNOWN	282.97
50	STUBE	40	50	1.0	1.0	NORM	0.	3.2879E-04	1.011	282.97	246.48
60	STUBE	50	3	1.0	1.0	NORM	0.	3.2879E-04	0.5204	246.48	263.13
200	MFRSET	3	200	1.0	1.0	NORM	0.	5.0000E-04	-288.8		
100	STUBE	3	100	1.0	1.0	NORM	0.	1.9708E-05	0.3075	42.060	42.012

SUBMODEL NAME = RAD

ARLXCA =	1.000000E-02	ARLXCC =	0.	ATMPCA =	1.000000E+30	ATMPCC =	0.	BACKUP =	0.		
CSGFAC =	1.00000	CSGMAX =	102.385	CSGMIN =	9.58313	DRLXCA =	1.000000E-02	DRLXCC =	0.		
DTIMEB =	1.000000E+30	DTIMEI =	0.	DTIMEL =	0.	DTIMEU =	0.	DTMPCA =	1.000000E+30		
DTMPCC =	0.	EBALNA =	0.	EBALNC =	1.251221E-03	EBALSA =	1.000000E-02	EBALSC =	0.		
ESUMIS =	414.938	ESUMOS =	414.936	EXTLIM =	50.0000	ITBOLD =	10	NARLXN =	0		
NATMPN =	0	NCGMAN =	40	NCSGMN =	10	NDRLXN =	0	NDTMPN =	0		
NEBALN =	0	NLOOPT =	100	ITEROT =	0	ITERXT =	3	OPEITR =	0		
OUTPUT =	1000.00										

GLOBAL CONTROL CONSTANTS

ABSZRO =	0.	SIGMA =	5.670000E-08	TIMEM =	119.097	TIMEN =	120.000	TIMEND =	180.000
TIMEO =	120.000								

MODEL = SAMPLE FLUINT SAMPLE MODEL
FORWRD

FLUID SUBMODEL NAME = LOOP ; FLUID NO. = 7717

MAX TIME STEP = 17.8006 ; LIMITING TANK = 3 REASON = TEMP./DENSITY CHANGE LIMIT
LAST TIME STEP = 10.0000 VS. DTMAXF/DTMINF = 1.000000E+30 / 0. ; AVERAGE TIME STEP = 3.92603
PROBLEM TIME TIMEN = 180.000 VS. TIMEND = 180.000

LUMP PARAMETER TABULATION FOR SUBMODEL LOOP

LUMP	TYPE	TEMP	PRESSURE	QUALITY	DENSITY	ENTHALPY	HEAT RATE	MASS RATE	ENERGY RATE
1	TANK	300.0	1.0669E+06	0.5628	13.61	4.8182E+04	400.0	-3.6409E-07	0.1126
2	TANK	300.0	1.0667E+06	0.7956	9.690	3.1802E+05	0.	1.5570E-06	-0.8963
3	TANK	271.3	1.0667E+06	0.	639.9	-7.8156E+05	0.	-2.3037E-06	14.41
10	JUNC	300.0	1.0667E+06	0.4576	16.65	-7.3754E+04	-138.7	0.	0.
20	JUNC	300.0	1.0667E+06	0.1713	42.49	-4.0559E+05	-117.4	0.	0.
30	JUNC	300.0	1.0667E+06	1.0143E-03	556.5	-6.0299E+05	-69.86	0.	1.5259E-05
40	JUNC	276.6	1.0667E+06	0.	632.4	-7.4730E+05	-51.07	0.	-1.5259E-05
50	JUNC	265.5	1.0667E+06	0.	648.0	-8.1855E+05	-25.22	0.	-7.6294E-06
200	JUNC	275.9	1.0669E+06	0.	633.3	-7.5156E+05	15.00	0.	0.
100	PLEN	271.1	1.0667E+06	0.	640.1	-7.8240E+05	0.	1.1108E-06	-0.8682

MODEL = SAMPLE FLUINT SAMPLE MODEL
FORWRD

FLUID SUBMODEL NAME = LOOP ; FLUID NO. = 7717

MAX TIME STEP = 17.8006 ; LIMITING TANK = 3 REASON = TEMP./DENSITY CHANGE LIMIT
LAST TIME STEP = 10.0000 VS. DTMAXF/DTMINF = 1.000000E+30 / 0. ; AVERAGE TIME STEP = 3.92603
PROBLEM TIME TIMEN = 180.000 VS. TIMEND = 180.000

TIE PARAMETER TABULATION FOR SUBMODEL LOOP

TIE	TYPE	UA	QTIE	LUMP	TEMP.	NODE	TEMP.	PATH 1	FRACT	PATH 2	FRACT
10	HINC	46.81	-138.7	10	300.0	RAD.10	296.9	10	1.0000	20	0.5000
20	HINC	23.71	-117.4	20	300.0	RAD.20	294.8	20	0.5000	30	0.5000
30	HINC	5.487	-69.86	30	300.0	RAD.30	279.4	30	0.5000	40	0.5000
40	HINC	6.129	-51.07	40	276.6	RAD.40	267.8	40	0.5000	50	0.5000
50	HINC	6.426	-25.22	50	265.5	RAD.50	260.8	50	0.5000	60	1.0000

MODEL = SAMPLE
FORWRD

FLUINT SAMPLE MODEL

FLUID SUBMODEL NAME = LOOP ; FLUID NO. = 7717

MAX TIME STEP = 17.8006 ; LIMITING TANK = 3 REASON = TEMP./DENSITY CHANGE LIMIT
LAST TIME STEP = 10.0000 VS. DTMAXF/DTMINF = 1.000000E+30 / 0. ; AVERAGE TIME STEP = 3.92603
PROBLEM TIME TIMEN = 180.000 VS. TIMEND = 180.000

PATH PARAMETER TABULATION FOR SUBMODEL LOOP

PATH	TYPE	LMP 1	LMP 2	DUP I	DUP J	STAT	XL UPSTRM	FLOWRATE	DELTA PRES	FLOW REGIMES OR REYNOLDS NO.	
1	TUBE	200	1	1.0	1.0	NORM	0.	5.0000E-04	67.48	431.40	UNKNOWN
2	TUBE	1	2	1.0	1.0	NORM	0.5628	5.0036E-04	139.9	UNKNOWN	UNKNOWN
3	CAPIL	2	3	1.0	1.0	DLS	0.	1.4490E-04	65.38		
10	STUBE	2	10	1.0	1.0	NORM	0.7956	3.5390E-04	21.09	UNKNOWN	UNKNOWN
20	STUBE	10	20	1.0	1.0	NORM	0.4576	3.5390E-04	30.69	UNKNOWN	UNKNOWN
30	STUBE	20	30	1.0	1.0	NORM	0.1713	3.5390E-04	11.02	UNKNOWN	UNKNOWN
40	STUBE	30	40	1.0	1.0	NORM	1.0143E-03	3.5390E-04	0.9386	UNKNOWN	307.99
50	STUBE	40	50	1.0	1.0	NORM	0.	3.5390E-04	1.082	307.99	269.01
60	STUBE	50	3	1.0	1.0	NORM	0.	3.5390E-04	0.5555	269.01	287.95
200	MFRSET	3	200	1.0	1.0	NORM	0.	5.0000E-04	-272.7		
100	STUBE	3	100	1.0	1.0	NORM	0.	1.1108E-06	1.7103E-02	2.4102	2.4064

PROBLEM E: CAPILLARY PUMPED LOOP START-UP

In heat pipes, surface tension forces provide the pumping power necessary to circulate a working fluid between an evaporator and a condenser. A *capillary pumped loop* (CPL) works the same way, but the liquid flow is separated from the vapor flow for efficiency. A CPL resembles a simple mechanically pumped loop except that no pump is needed; the capillary structure in the specialized evaporators provides the pumping action. For this reason, the evaporators in CPLs are usually referred to as evaporator-pumps (EPs).

CPLs are currently being developed for spacecraft thermal transport systems. By avoiding mechanical pumps or control valves, CPLs have no moving parts and require little parasitic power. However, because of the dependence on capillary forces, they are sensitive to gravity and can transport only a limited amount of energy over a limited distance.*

This problem was chosen to demonstrate the following FLUINT features:

1. component models (CAPPMP)
2. capillary devices
3. initializing two-phase states
4. modeling with junctions
5. "half-length" modeling and associated accelerations
6. uses of heat transfer area fractions
7. duplication factors and dual one-way conductors
8. using restart options
9. overriding convective heat transfer calculations
10. modeling a real system

This problem is worked as both a detailed model and a simplified model. The simplified model demonstrates the use of heater junctions, RAPPR-generated simplified fluids, and other time-saving methods and assumptions.

---

* 24 kW over a distance of 10m has been demonstrated by the OAO Corporation.

** Most of the information in this example was extracted from the User's Manual for the CPL Modeler, Version 1.2, written by L. Nelswanger and R. Schweickart of NASA–GSFC, and J. Ku and E. Itkin of OAO Corporation. Some data were estimated. The author would like to thank Mr. Schweickart, for his extra assistance in preparing this model.

E.1 Problem Description

A prototype aluminum/ammonia CPL was flown in a space shuttle Getaway Special (GAS canister) experiment.** The CPL (Figure E–1) consisted of two EPs, a temperature-controlling reservoir, and a combined condenser/subcooler. The fixed-volume reservoir was cold-biased and used a heater to maintain a constant temperature. The condenser piping was embedded in a radiator plate as shown in Figure E–2. This "piping" was actually an axially-grooved heat pipe extrusion, shown in cross-section in Figure E–3.. The radiator plate was mounted to a 65 kg aluminum sink plate using a thermal bonding material. The sink plate could radiate only a limited percentage of the total CPL throughput; the rejection capability was purposely undersized to investigate performance under varying rejection temperatures.

The evaporator consisted of a tubular polyethylene wick. Liquid was supplied to the inner diameter with evaporation occurring at the outer diameter of the wick, where the wick was in contact with the tips of 45 grooves running axially along the inner pipe wall. The evaporator extrusion, like the condenser, was a trapezoidal groove heat pipe cross-section. The "isolator" used virtually the same wick inside of the liquid header. The purpose of an isolator is to prevent deprime (or loss of capillary pumping) in one EP from causing deprime in the adjacent EP.

In one experimental run, the entire system was initially resting at 6.6°C with the exception of the reservoir which was held at 29°C. At time zero, 255 Watts were applied to the outside of each EP. The sink plate began to warm up, and the temperature increased linearly from 6.6°C to about 17°C by the end of the experiment, 42 minutes later. The goal of this problem is to simulate this experimental run.

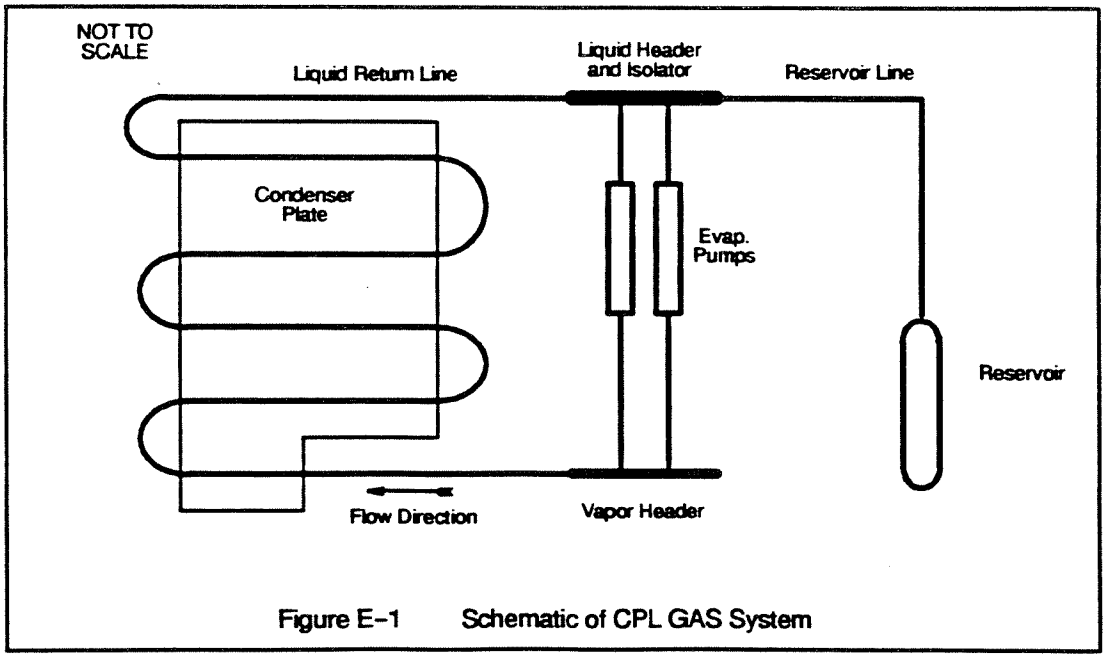


Figure E–1 Schematic of CPL GAS System

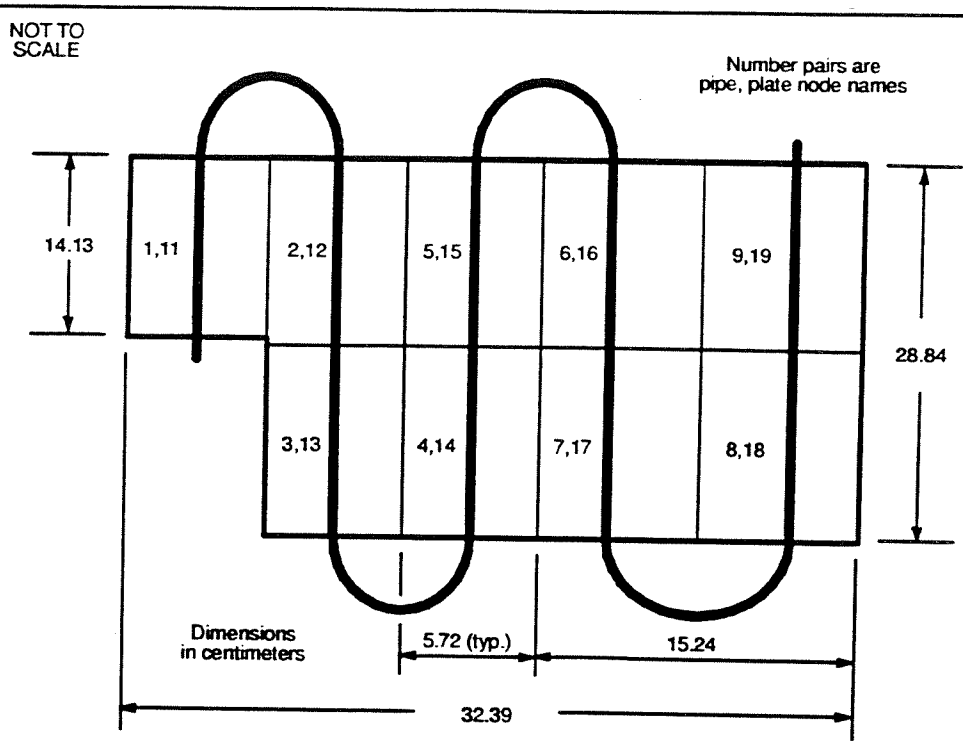
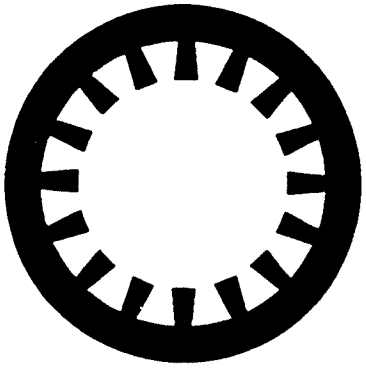


Figure E-2 Condenser Plate Size and Node Names



Outer dia:	7.94 mm
Inner dia:	6.43 mm
Vapor core dia:	4.70 mm
No. of grooves:	16
Cap. radius:	0.61 mm
Tot. Grv. Area:	9.9 mm^2

Figure E-3 Cross Section of Condenser Extrusion

The parameters for this CPL system are listed below:

Reservoir:
- Reference (saturation) temperature ..... 29°C
- Connecting line length ..... 70 cm
- Connecting line diameter ..... (see condenser)

Evaporators:
- Number 2
- Active length ..... 8.89 cm
- Power input ..... 255 W
- Thermal capacitance of an EP ..... 137.7 J/K
- Radial conductance through pipe ..... 158.6 W/K
- Evaporative heat transfer coefficient ..... 2500 W/m^2-K
- Number of axial grooves per EP 45
- Groove hydraulic diameter ..... 1.4 mm
- Groove flow area ..... 1.5E-6 m^2

Evaporator inlet/outlet lines:
- Liquid supply line length ..... 15.24 cm
- Liquid supply line ID ..... 4.572 mm
- Vapor exhaust line length ..... 7.94 cm
- Vapor exhaust line ID ..... 6.604 mm
- Vapor exhaust line capacitance ..... 5 J/K

Wick (evaporators and isolators):
- Wick ID ..... 1.113 cm
- Wick OD, evaporator ..... 2.286 cm
- Wick OD, isolator ..... 2.134 cm
- Pumping radius ..... 16.5E-6 m
- Permeability ..... 2.6E-13 m^2

Headers (liquid header contains isolator):
- Vapor header length ..... 12.7 cm
- Vapor header ID ..... 1.41 cm
- Vapor header capacitance ..... 10 J/K
- Liquid header length ..... 12.7 cm
- Liquid header ID ..... 2.134 cm

Condenser line (axially grooved):
- Total length ..... 1.57 m
- Number of axial grooves 16
- Vapor core diameter ..... 4.70 mm
- Total wetted perimeter ..... 47.1 mm
- Total flow area ..... 27.2 mm^2
- Total groove flow area ..... 9.9 mm^2
- Capillary pumping radius ..... 0.61 mm
- Tube ID to plate conductance (per tube length) ..... 1.19 W/cm-K
- Tube capacitance (per length) ..... 0.547 J/cm-K

Line from condenser to liquid header:
- Total length ..... 15 cm
- Line ID ..... (see condenser)

Condenser plate:
- Capacitance per unit area ..... 2.85 J/cm^2-K
- Aluminum thermal conductivity ..... 165 W/m-K
- Thickness ..... 1.27 cm
- Conductance to sink per unit area ..... 0.32 W/cm^2-K

E.2 A SINDA/FLUINT Model

The event to be simulated is almost purely a radiator transient. The start-up transients last less than one minute and can be neglected compared to the length of the experiment. For this reason, the analysis could be started by assuming that the evaporators and headers are already cleared of liquid and that vapor is just beginning to arrive in the condenser. As will be shown later, this assumption leads to perfectly valid results in a short time. However, *for demonstration purposes,* this analysis will include the start-up transient and the extra complexity and computational time needed to resolve such events. Good modeling practices would suggest dropping this level of detail unless the start-up itself were the purpose of the analysis. (Section E.5 contains a description of a greatly simplified model for comparison purposes.)

As the first modeling decision, a plenum is used to model the reservoir. This decision is forced by the lack of data concerning the reservoir, but would probably be a valid choice in any case. This modeling choice is equivalent to assuming perfect temperature control in the reservoir. The plenum will be assumed to remain as saturated liquid at 29°C. Other modeling choices for the reservoir include a single tank or a coupled pair of tanks (one for the vapor, one for the liquid). When cold liquid dumps into the accumulator during start-up, the behavior of a single tank may be far from reality due to the perfect mixing (equilibrium) assumption: the tank pressure will drop too fast. In actuality, the liquid entering the reservoir tends to compress and heat the vapor, initially raising the pressure until the two phases return to an equilibrium pressure below the initial value. Given enough reservoir design and initial condition information, a nonequilibrium model could be developed using the NONEQ0 and NONEQ1 routines, as shown in Sample Problem F.

No volume is attributed to the reservoir connecting line since this does not take part in the recirculation of the fluid. A K-factor of 1.0 is added to this line because of exit losses into the reservoir. These losses are only important during the fast purging of the liquid in the system upon start-up.

Because of the relatively small volume in the condenser and because the system response is expected to be dominated by the response of the sink, junctions and STUBE connectors will be used to model the condenser. Nine segments are adequate to represent the radiator/plate system. Centered discretization is used not only because it is recommended for two-phase flow, but because it facilitates tying to thermal nodes centered in each plate section (Figure E–2.) An additional STUBE connector is placed at the outlet of the condenser to represent the liquid return line. Because of the long radius bends and other smooth transitions, additional head losses are small. K–factors of 0.4 are attributed to the bends and 0.37 to the inlet.

Because of the presence of axial grooves in the condenser piping, the modeling decisions surrounding the condenser itself deserve significant discussion. These grooves have the effect of preferentially trapping and redistributing liquid, maintaining a relatively constant void fraction*. Detailed simulations of the separate passages were performed but did not yield significantly different results over those produced by a very simple treatment: a hydraulic diameter

---

* Because a relatively large pumping radius implies minimal pressure differences between phases (at least compared to axial gradients), only one lump is needed to represent both phases: parallel LINE or HX macros are not needed for grooves and the vapor core. This would not be true in a heat pipe, where the axial pressure gradient must be less than the surface tension gradient. This does not preclude the use of twinned paths along a single row of lumps, but such a model requires more effort than the results justify in order to be faithful to the noncircular geometry (i.e. the user must calculate appropriate values for FD, FC, etc.).

assumption. The grooved cross section and the existence of two-phase flow certainly make the use of this assumption questionable: the separation effect of the grooves is not included and the resulting hydraulic diameter is probably too small even for single-phase flow. A larger hydraulic diameter could be used for pressure drop purposes, and then tie area fractions could be increased appropriately for heat transfer calculations (which back out the wetted perimeter and heat transfer area based on 4.0*AF/DH). However, the hydraulic assumption appears to be adequate in this case because the pressure drop through the condenser is irrelevant except during the initial liquid purge event, in which case the radiator contains liquid initially and the evaporator-pump is in danger of deprime due to condenser resistance.

More important is the impact on heat transfer calculations. Use of the standard annular condensation correlation ROHSEN (implied by the use of HTN or HTNC ties in an HX macro) would predict axially decreasing heat transfer coefficients as the liquid layer builds up, whereas in the actual case liquid is redistributed somewhat by the grooves. To mimic this redistribution, the actual heat transfer coefficient will be an average of a constant (based on groove heat transfer) and what ROHSEN predicts. The net effect is a decrease of the heat transfer coefficient at the inlet of the condenser over a plain tube, and an increase at the end of the condensation zone.

Kamotani's correlation for grooved condensation heat transfer is:

$$H = \frac{N_G K_L}{0.0221 + \frac{K_L t}{K_w w}}$$

where:

- H ..... heat transfer coefficient
- K_L ..... liquid conductivity (0.54 W/m-K at 270K)
- K_W ..... wall conductivity (165 W/m-K)
- N_G ..... groove density (about 1080 per meter)
- t ..... groove thickness (height, about 0.86 mm)
- w ..... groove width (at interface, about 0.36 mm)

A quality of 0.025 corresponds to completely filled liquid grooves and a dry vapor core. Above this quality (e.g., nearly all of the two-phase region in the condenser), a constant groove heat transfer coefficient will be averaged with ROHSEN. Below this quality, the heat transfer coefficient must blend smoothly into the a single-phase convection value as the central core fills. This could be accomplished with HTU ties whose UA is updated in FLOGIC 0 in a Fortran DO LOOP. However, this would create an instability at low qualities if the UA were predicted as an *explicit* function of junction quality, which is an instantaneous parameter. If the quality is currently 0.0, a low single-phase UA might cause the quality to jump up near that of the upstream quality. If the quality is currently above 0.025, a high two-phase UA might drive the quality down to 0.0.

There are several ways to deal with this instability. First, the user might add heavy damping to the UA prediction such as by using a weighted average of the new and old UA values. Such damping is relatively easy to implement but has hidden drawbacks such as slowing the solution or artificially retarding the condenser response, and can actually *cause* instabilities if done improperly. Second, the user might base the UA prediction on the quality of the upstream lump (perhaps using the XTRACT subroutine to simplify the DO LOOP logic). This method is

very stable and easy to implement, but alters the results slightly. Third, the user can *implicitly* predict the UA and the corresponding junction quality. This method is stable but difficult to implement since relaxation iterations are often required. Fourth, the user can alter the basic condensation routine ROHSEN to include groove heat transfer correlation and place the alternate version in SUBROUTINE DATA. This method, which is used in this sample problem, causes the replacement routine to be linked preferentially over the standard version in the processor library. This replacement method takes advantage of internal FLUINT logic that performs the implicit solution mentioned in the third option, and has the additional benefit of improving the low quality (slug flow) heat transfer coefficient based on scaling factors inherent in the ROHSEN routine. However, the main drawback to this method is that condensation heat transfer would then be altered for all HTN and HTNC ties in all fluid submodels. Therefore, special logic is added into the new ROHSEN routine to avoid groove heat transfer correlations unless the flow area matches that of the condenser: 2.72E-5 m^2. The normal ROHSEN correlation then applies to the evaporator outlet header.

Kamotani's correlation predicts 19,400 W/m^2-K at 270K. When used as an ammonia heat pipe condenser, the heat transfer coefficient for the extruded section was determined experimentally to be 11,800 W/m^2-K at 270K, based on vapor core diameter. As will be seen in later sections, it is not immediately clear which value should be used. Analyses were performed with both coefficients (the fixed heat pipe value was scaled to higher temperatures by assuming proportionality with liquid conductivity). In all cases, predicted temperatures are too low, perhaps due to other causes such as overestimation of plate capacitance or underestimation of thermal conductances to the base plate. Therefore, higher values of the two-phase heat transfer coefficient cause a better match at the inlet of the condenser at the price of a worse match at the outlet. Higher coefficients also affect run time by decreasing the thermal time constant CSGMIN. In this sample problem Kamotani's correlation is used. The reader is encouraged to try the lower experimental value (11,800 W/m^2-K) as well.

With regard to the thermal model of the condenser plate, the nodalization will correspond to that of the condenser piping. The distinction between the tubing and the plate warrants separate nodes because of the importance of the correct wall temperature on the two-phase heat transfer. Each tube node will conduct energy to a corresponding plate node. Even though their mass is relatively small and they will be adjacent to the relatively large conductance resulting from two–phase heat transfer, the pipe nodes will be modeled as diffusion nodes for illustration purposes. In the simplified model that follows, significant savings will be demonstrated by reversing this decision and using arithmetic nodes instead.

Figure E–2 contains the names and locations of the plate and tube nodes and lumps. The plate nodes will conduct energy to adjacent plate nodes as well as to the sink. Fin efficiencies of 63% and 51% are attributed to the plate-to-sink conductances since the plate nodes exist at the root of each "fin." The sink itself is modeled with a boundary node whose temperature is updated linearly with time.

The distance between the two EPs is small in relation to the diameters of the headers; the pressure differentials between these two points will be completely negligible. Thus, one tank will be used to model each header. A higher fidelity model could be built using two tanks per header, connected by a STUBE connectors representing the header itself, but this would be implicitly ask the program to split analytic hairs. More importantly, this simplifying assumption

permits symmetry to be exploited. One EP will be modeled, and duplication factors at the inlet and outlet of the headers will be set to 2.0. As will be seen, the entire model (including the thermal portion) must be consistent with this time-saving symmetry.

An inlet tank (subcooled liquid) will contain the volume of the inlet line and the liquid header, as well as the volume in the condenser. An adjacent tank will represent the volume inside *one* evaporator pump. These two tanks are separated by a CAPIL connector that represents the isolator. While the vapor-barrier effects should not be important in this example, the pressure-flowrate behavior of a wick is easily taken into account with this device. (The only other FLUINT connectors that exhibit a linear loss are a NULL connector and an STUBE connector with FPOW=0.0.) An upstream duplication factor (DUPI) of 2.0 is applied to the CAPIL connector. This means that two such passages exist from the point of view of the inlet header. Models of the header itself and the inlet tubing are ignored because of negligible pressure losses in these elements compared to the wick losses.

The EP pumping action can be modeled with the CAPPMP component model. A tie will be made in this macro to a node representing the evaporator wall, with the UA value being based on a number estimated from experimental results. The upstream end of this CAPPMP will be connected to the tank representing the liquid side of the EP. Also, a tank is needed to model the outlet (vapor) header.

Many modeling options are possible to connect the CAPPMP with the outlet header. First, they may be connected directly, ignoring any heat transfer or losses associated with the grooves or the outlet plumbing. This would be appropriate for a model that did not focus on the start-up events. For the purposes of this model, extra lumps and paths will be used to represent the grooves and the outlet line.

The choice of tanks versus junctions for these evaporator outlet lumps is important. Junctions are preferred for several reasons in this particular example. First, the energy and time it takes to clear liquid out of these small volumes is not as important as it might seem. Both are proportional to the mass of *vapor* in them, not liquid! Almost all of the liquid initially in these lumps will be *displaced* by generated vapor rather than *evaporated*. Also, vapor tanks are sensitive to cold shocking in such applications: if the pressure on the liquid side became larger than that on the vapor side, an unrealistically sudden collapse of the vapor tank pressure would occur. Therefore, junctions will be used. The only reason one might wish to use a tank for the grooves is to allow the CAPPMP to reprime more realistically. If the CAPPMP deprimes, the flow reverses immediately in the vapor junction and the quality may instantly become too small to allow reprime. This tendency will be offset by the movement of the HTM tie, which will move to the vapor junction in the event of a deprime. This outlet junction is the vapor side by default. To avoid the default vapor side, the VAPOR keyword should be used.

A call to SPRIME will be used to clear the grooves when the wall temperature exceeds saturation. This call will start the pumping action in the CAPPMP. Once the pumping action starts, vapor will start to flow immediately into the subcooled outlet header, where it will initially collapse. This logic would not be necessary if a tank had been used or if the initial flowrate were nonzero. Because there is no initial flow in the junction, the presence of the deprimed CAPPMP tie on it has no effect: heat cannot be added into a junction that has no flow through it. No flow is present until the CAPPMP has primed—the model must be bootstrapped into action.

An STUBE will be used to represent the flow in the EP vapor grooves. It will represent a single groove, and will have duplication factors of 45 on both ends. (From the point of view of the entire system, 90 such grooves exists. This is an example of exploiting a symmetry within a

symmetry.) Because vapor is generated along the length of these grooves, the flowrate is zero at one end and increases linearly toward the exhaust end. Therefore, the STUBE will be modeled at only half the actual length, and will carry the full flowrate. This is the "half-length" method described in the main volume, Section 4. The acceleration of the fluid due to the radial injection is taken into account with an AC factor that will be updated in the FLOGIC 0 block. This optional acceleration results in approximately 20% added pressure drop in this case. A similar factor applies to the vapor header, although in this case the factor is an approximation because the "injection" is not uniform.

A thermal model is needed to describe the evaporator and outlet header structure. (Similar models of the inlet structure are neglected because of negligible temperature changes and small single-phase heat transfer coefficients.) Three nodes are needed for the evaporator wall, the exhaust line, and the vapor header wall. To avoid small diffusion nodes, the exhaust line will be an arithmetic node and its mass will be lumped into the adjacent header node. Small axial conductors are added between these nodes. The addition of such conductors, while having a negligible impact on system response, has the benefit of complying with a SINDA requirement that no node be thermally isolated even if it is tied to a fluid lump. Otherwise, a single node could have been used. Because two exhaust lines exist from the point of view of the header, dual one-way conductors will be used for that connection. This is the SINDA equivalent of the FLUINT duplication factors, and could be avoided by the use of tie duplication factors.

A fluid tie will be made between the exhaust line node and the junction inside. This is the reason for the use of an STUBE connector for this line even though the pressure drop is negligible: it facilitates heat transfer calculations. Similarly, a tie is made between the header wall node using an STUBE connector representing the half-length of the header. Because of the half-length assumption, an area fraction of 2.0 is applied to the tie to return the heat transfer area to its full value. An additional estimated factor is applied to increase the heat transfer due to the impingement of the exhaust flow on the opposite wall of the vapor header and the related increased mixing of the fluid. Otherwise, a fully-developed convection correlation would underpredict the heat transfer coefficient. Although the acceleration in the header is included, the real purpose of the STUBE is to facilitate heat transfer calculations, not to accurately model the small pressure losses in the header.

Surprisingly, the behavior of the exhaust line and header walls is critical to the successful start-up of the loop. These provide the thermal inertia required to *slowly* clear the vapor header of liquid. With adiabatic boundary conditions in this part of the loop (or too-warm walls or too rapid heating), the CPL deprimes while clearing fluid out of the vapor header due to the large transient resistance through the condenser. A cold wall helps condense some of the excess vapor from the evaporator, giving the header time to empty.

The names and types of the network elements composing the total model are summarized below, and shown schematically in Figure E–4:

CPL FLUID SUBMODEL:

PLENUM 99	Reservoir
STUBE 99	Reservoir connecting line, plus exit loss into reservoir
TANK 10	Inlet (liquid) header
CAPIL 300	Isolator wick
TANK 20	Liquid inside of EP
CAPPMP	EP
JUNCTION 1000	Center junction
"NULL" 1000,2000	Inlet and outlet specialized connectors
JUNC 30	Evaporator groove
STUBE 30	Evaporator groove, half length
JUNC 35	Evaporator outlet, exhaust line
STUBE 35	Vapor exhaust line
TANK 40	Outlet (vapor) header
STUBE 40	Vapor header line, half length
JUNCTION 50	Condenser inlet
JUNCTIONS 1-9	Condenser/subcooler
STUBES 1-10	Condenser/subcooler, plus contraction and bend losses
JUNCTION 100	Condenser outlet
STUBE 100	Liquid return line

EVAP THERMAL SUBMODEL:

NODE 1000	Evaporator wall
NODE 1500	Exhaust line wall, arithmetic
NODE 2000	Vapor header wall, contains mass of #1500
CONDUCTOR 1000	Evap to exhaust (both ways)
CONDUCTOR 1500	Exhaust to header (one way, double value)
CONDUCTOR 1501	Header to exhaust (one way, single value)
CONDUCTOR 2000	Header to plate inlet (both ways)

PLATE THERMAL SUBMODEL:

NODES 1-9	Condenser/subcooler piping
NODES 11-19	Condenser plate
NODE 300	Sink (boundary)
CONDUCTORS 1nn	Tube to plate
CONDUCTORS 3nn	Plate to sink
CONDUCTORS 2nn	Plate to plate, perp. to flow (tube to tube)
CONDUCTORS 4nn	Plate to plate, flowwise (axial)

E.3 The Input File

The input file is provided on the following pages.

To demonstrate restart options, this analysis is performed in two parts. The first run covers the first two minutes of real time, after which a restart file is produced. The second run starts by reading in the restart file and finishing the last forty minutes of analysis. Only the input file for the first run is shown; the input file for the second run can be generated from this file by commenting out two statements and uncommenting two others, as shown in the first two header blocks of the input file.

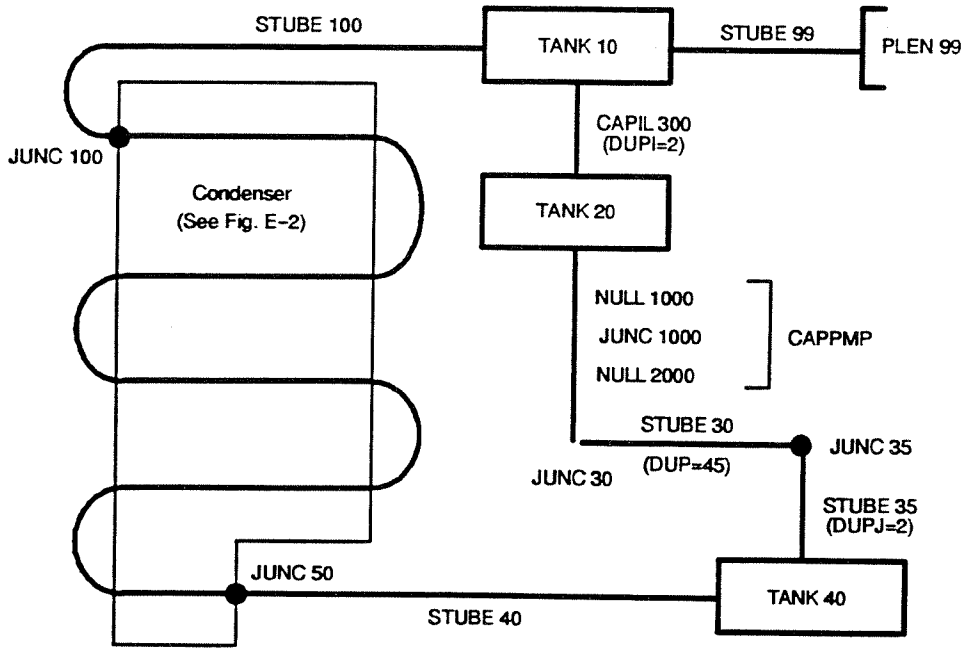


Figure E-4 Schematic of FLUINT Network

Restart options are often used to save steady-state results, or as in this case, to break up a large run into smaller parts to avoid losing results due to machine failures. When restarting FLUINT models, *all* parameters must be saved and replaced. Only global user data is never replaced to aid logical control of restarts. The basic restriction behind restarts is that the model must be the same as that saved. No elements, user data, or arrays can be added or deleted (although values can change).

OPTIONS DATA—As with previous examples, a summary file is written to USER1. Also, a QMAP file is named to contain output from an optional FLOMAP call. For the first run, a restart output (RSO) file is named. This file is used as a restart input (RSI) file for the second run.

CONTROL DATA—The unit system will be SI, with temperature in units of degrees Celsius. The problem will run for 42 minutes, with output produced at 6 second intervals for the first run, and then two minute intervals thereafter (in the second run). Because a single thermal model OUTPUT CALLS will be used, the fluid model output interval, which is required, is set to a large number. A maximum time step of 10 seconds is allowed.

USER DATA—Two variables (TSAT and IFLAG) are used to mediate the initialization of the CAPPMP when the evaporator wall temperature exceeds the saturation value for the first time. The evaporation UA will be initially zero, and then replaced with UAEVAP when the CAPPMP is started. Note that single-phase heat transfer is neglected since the system flowrate will be negligibly small until the CAPPMP starts pumping.

The NTEST variable is used to control the restart logic. For the first run, NTEST is zero. The tail end of the output for the first run lists record or "key" 121 as being written. This value is then used for NTEST during the second run to both signal an available restart file and to "unlock" the appropriate event saved in that file.

FLOW DATA—The default is set to be subcooled liquid at 6.6°C. The pressure could be estimated and and then overwritten later (via CHGLMP calls in OPERATIONS DATA) to exactly correspond to saturation at 29°C, but instead the PL! option is used to mandate the desired pressure, which was determined by previous runs but could have been adequately determined from saturation tables. The default diameter and flow area correspond to that of the grooved condenser, which is the most common pipe size used. The initial flowrate is assumed to be zero.

A small compliance is used in the inlet tank (#20) to avoid unrealistic pressure spikes in an otherwise fixed-volume liquid tank. Such a compliance is optional in the tank located at the end of the condenser (Tank 10). Note that the vapor header compliance is larger than those of the other tanks. This helps prevent unrealistic surges in pressure when the tanks first starts to boil and vapor displaces liquid. Ordinarily, this surge lasts only a few time steps until the rest of the system can adjust. However, in a capillary model such surges may cause the device to deprime.

The evaporator section is defined first, including the ties to the corresponding thermal nodes. The upstream end of the CAPIL connector is assigned a duplication factor of 2.0 (DUPI), while the downstream end of the CAPPMP macro (DUPJ) is assigned the same value. This means that adding 255W to the CAPPMP junction is equivalent to adding 510W to the whole system. Note that in the calculation of CFC for the isolator and the evaporator wick, the logarithmic form of the "conduction" equation is used. The CFC through a cylindrical wick is given in the manual as $2 \cdot \pi \cdot P \cdot L / \ln(r_o/r)$, where P is the permeability, L is the length, and r_o and r are the outer and inner radii of the tube, respectively.

When the CAPPMP is defined, an HTM tie is made to the evaporator wall node. The UA value is zero until the wall temperature exceeds saturation and evaporation can commence. This tie will actually exist as an HTU tie on junction 30, the default vapor side, until the CAPPMP primes. The tie will then jump to the CAPPMP junction and become an HTM tie. In either case, the user may reset the UA value. Note also that XVH and XVL are set below the default values. This makes the CAPPMP more resistant to deprime caused by liquid in the outlet.

The STUBE representing the vapor grooves is assigned one half of the real length because of radial injection, as is the header STUBE. Similar methods could have been used in the liquid header and the liquid passage inside the EP wick if these had been modeled.

The condenser/subcooler is generated with a centered HX macrocommand. Note that the last condenser leg (the subcooler) is spaced farther apart than the other legs. This difference is neglected to facilitate input using one HX command. K–factor losses are to be added for the inlet contraction and the bend losses, but these cannot be added to the HX command or they would be applied to every generated connector. Instead, the FK values are initialized later in OPERATIONS DATA for selected connectors in the macro.

The initial state of the condenser is subcooled liquid. Being composed of junctions, the state inside the condenser will change instantly when the flow starts moving. Recall that DH and AF must be explicitly defined in an HX or LINE macro via the DHS and AFS input keywords.

The reservoir will exchange only saturated liquid with the system. While the plenum could be initialized in that state by setting a zero pressure and XL=0.0, here the quality is set to 0.1 and liquid suction is used on the connecting line. The RLS option is used since the plenum is on

the defined downstream end of the line. Alternatively, the positive flow direction could be chosen as *from* the plenum, and the LS option would then be specified. A K–factor of 1.0 is added to the STUBE connector to represent the exit losses into the reservoir.

NODE DATA—In the PLATE submodel, the sink and the tube and plate nodes are generated. Note that the difference is capacitance between tube nodes was neglected, while the capacitance of the different sizes of plate nodes was input faithfully. In the EVAP model, the three evaporator and outlet wall nodes are defined. The initial temperatures are all 6.6°C.

CONDUCTOR DATA—In the PLATE submodel, the differences between the tube node locations are neglected for the conductances to the plate, while the plate-to-plate and sink-to-plate conductors take size differences into account. The axial tube conduction, while almost negligibly small, is added into the axial plate conduction term. In the EVAP submodel, small axial conductors are used, one of which connects to the inlet of the condenser. Note the use of dual one-way conductors to be consistent with the symmetries exploited in the adjacent fluid submodel.

FLOGIC 0—The acceleration factor for the half-length EP grooves and the half-length vapor manifold are updated here according to the current density in the grooves. The subsequent logic is used to "kick start" the CAPPMP when the wall temperature exceeds saturation. DTMPCA is used to prevent the thermal model from stepping too far past this important transition. The minimum allowable time step is then adjusted to allow smaller time steps during the sudden purge event, but to require larger time steps at other times. Time steps cannot be specified: DTMINF simply states the minimum affordable time step below which the program should produce output and halt.

FLOGIC 1—In case of difficulties in modeling, preventative logic is added here to detect deprime in the EP. When a CAPPMP is primed, the GK values of the NULL connectors are zero. All such FLUINT "set-up" calculations, such as the updating of junctions, lump QDOTs, and connector GK and HK values are performed between FLOGIC 0 and FLOGIC 1. This fact is exploited here in combination with a check on the UA value. If the CAPPMP has just deprimed, a FLOMAP and other outputs are written to help the user debug the inputs, and FLOGIC 2 will terminate the run. Such preventative measures are invaluable in the process of arriving at a reasonable model.

FLOGIC 2—If the CAPPMP deprimes, the run will be terminated as signaled by resetting TIMEND. TIMEND should not be reset before the completion of the time step to avoid nonpositive time steps.

VARIABLES 1—The only calculation performed here is the update of the sink temperature as a function of time.

OUTPUT CALLS—Lump, path, and tie tabulations are written each output interval along with nodal temperatures. The summary file contains the temperatures for the first 7 pipe nodes, along with pressure gain and flowrate information. After two minutes of simulation, the output interval is increased to two minutes.

OPERATIONS DATA—After building both the thermal submodels and the fluid submodel as one configuration, the first order of business is initialize the condenser FK values. Second, NTEST is checked to see if this is the first or second run. If a restart run is available, the results previously saved from the first run are loaded back into the program. Output intervals and time step controls are *subsequently* readjusted to reflect the relatively calm network that can be expected after the system has started. Note that if these constants had been reset *before* the call to RESTAR, the values would have been overwritten by the RESTAR call. Because named user constants are not saved or restarted, IFLAG must be manually reset to signal that the CAPPMP has already started. For the same reason, the initialization of TSAT need only be performed once. TSAT will be very close to 29°C. When the header for the user file has been written, FWDBCK is called to start the transient integration (or finish it, in the case of the second run). If this is the first run (signaled by zero NTEST), results are saved in the RSO file for use in the second run.

SUBROUTINE DATA—The replacement version of ROHSEN was copied from the processor library, modified to include Kamotani's grooved heat transfer correlation in the special case where the flow area exactly matches that of the condenser extrusion (and the quality is not too much to flood the central passage), and appended to the input file as SUBROUTINE DATA. This file could also be INCLUDEd. DEBOFF and FSTOP commands are used before this routine to turn off the debugger and the translator, respectively.

E.4 Output Description

The user summary files and selected printings from the OUTPUT files of the two runs are provided on the pages following the input listings.

As a preliminary remark, note that the difference in pressures between the CAPPMP lumps does not include the pressure drop though the wick itself when the device is primed. This pressure drop is taken into account internally by the CAPPMP model to determine the pressure differential at the liquid/vapor interface.

With regards to the start-up transient (see Figure E–5), the CPL primes and begins to pump after about 12 seconds. The vapor header reaches saturation and begins to empty after about 20 seconds. The system has nearly achieved a hydrodynamic steady-state after about 30 seconds, which is clearly negligible compared to the total experimental run if the user is simply interested in overall thermal response. In fact, *much of the computational cost of this run is incurred between 20 seconds and 30 seconds,* during the relatively explosive clearing of the vapor header.

However, the events that occur within the first few seconds are very important from the viewpoint of a CPL designer. As noted before, if the user neglects the thermal inertia in the plumbing at the outlet of the evaporator, the CPL will deprime because it is unable to provide sufficient head to transiently clear the outlet of liquid. In other words, to avoid excessive pressure rises in the outlet, the *volumetric* flowrate of the saturated fluid that enters the condenser (flowing to the reservoir) must equal the *volumetric* flowrate of the saturated vapor being generated in the EPs. This implies a substantial transient flowrate through the condenser, roughly equal to the mass flowrate through the evaporators (~4.0E-4 kg/s) times the ratio of liquid to vapor densities (~600/8.8 = 68) for a total transient flowrate of approximately 2.8E-2 kg/s. Even neglecting the inertia in the condenser and reservoir lines, the static resistance results in a

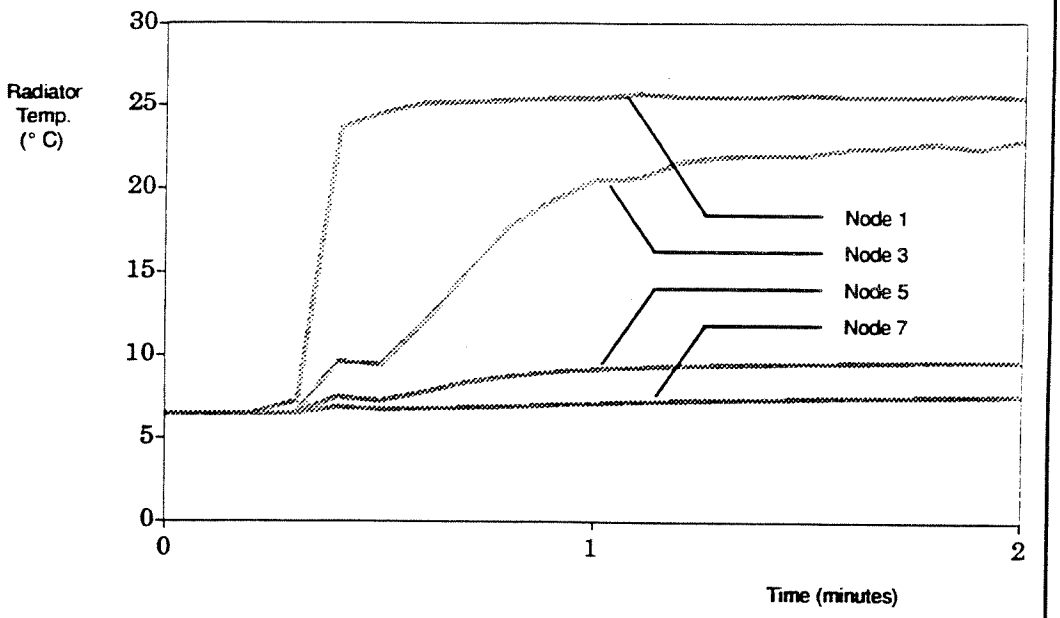


Figure E-5 Condenser Pipe Wall Temperatures: First Two Minutes

pressure gradient several times greater than the pumping provided by the wick. Hence, deprime occurs. (The EP may subsequently reprime, although modeling such behavior requires a tank rather than a junction to represent the grooves.) When the heat transfer to the cold outlet plumbing is included, this purge event is spread over time, reducing the peak flowrate to a manageable level: approximately 7.0E-3 kg/s or one fourth of the peak adiabatic value.

This implies that there is a limit on the power that can be applied to a flooded CPL, and that this limit is lower than the ultimate steady-state power limit. Furthermore, this limit decreases substantially in a CPL with negligible mass in the outlet or with initially warm walls, or with a large flow resistance in the condenser. It should be noted that other phenomena help overcome this transient limit, and that the CPL may partially rather than completely deprime. This limit is therefore not encountered very often in real CPL systems.

Another interesting phenomena occurs during this fast purge event. When the two-phase mixture starts to enter the still-cold condenser in these first few seconds, the vapor is rapidly condensed in the first condenser segment. As a result, high velocity, low density flow is rapidly decelerated to a lower velocity, higher density flow, causing a pressure rise in the condenser. Once the system settles down, this deceleration rise is equal to about 60 Pa (~0.01 psid). However, during the purge event, the deceleration pressure rise is on the order of 7000 Pa (~1 psid). This rapid, extreme condenser pumping can artificially flood (and therefore deprime) the evaporator pumps unless all of the losses in the system are included and realistically represented. As noted before, too many losses in the condenser can result in deprime during the initial liquid purge, yet too few losses can result in flooding later as the two-phase mixture enters the cold condenser. Careful modeling is required to accurately estimate the behavior of all loss mechanisms.

It should be apparent that the start-up of the CPL, modeling in detail, is not as trivial as it might first appear. The cost of including such detail should be apparent to the reader (see also Section E.5 for a comparison with a simplified model; note the cost of the first run covering the start-up is a significant fraction of the total cost even though the second run covers twenty times as much real time). What might not be apparent is that *the time and effort required of the user is often proportional to the computational time required by the program.* This may contradict some user's intuition that the more work left to the computer, the less work the user must perform. However, this is a "toolbox" code, meant to be scalable to various levels of detail by exploiting the user's knowledge, available information, and expertise to simplify the problem. *The more detail the user requests and the fewer assumptions he makes, the more data he must supply, and the more the accuracy of this supplied data becomes critical.*

Most of the interesting events during this start-up event are not evident because they all happen within one output interval. A common mistake in reviewing outputs is to neglect the sampling errors inherent in the requested output interval: the resulting profiles are "connect-the-dot" simplifications of the real profile, and important maxima and minima can easily be excluded.

After the start-up, the flowrate gradually increases because the temperature at the inlet of the EPs gradually rises. The CAPPMP model does not normally include any appreciable superheating of the vapor; 255W accomplished more boiling and less sensible heating as the inlet temperature approaches saturation.

Figure E–6 displays the long term temperature histories for selected pipe wall nodes, along with the bracketing flight data. The first pipe segment contains the most vapor flow and so heats up quickly, whereas the last segment only starts to enter the two-phase regime at the end of the run.

The latter part of this analysis is essentially a test of the condensation heat transfer correlation, and of the values of capacitance and conductance in the thermal model of the condenser. Overall, the comparison is adequate—the trends are the same over the same time scales even though the lines do not lie on top of each other. Like the flight article, segment 7 just begins to enter the two-phase region 42 minutes after heat is applied. However, the simulation consistently falls about 2°C short of the flight data, especially at the inlet of the condenser (segment 1).

This difference is due either to overestimation of the pipe-to-sink conductance, overestimation of the pipe/plate capacitance, significant superheating, or to an effective reservoir pressure that is higher than the initial value (saturation at 29°C), or perhaps combinations of all four. Evidence for the last two reasons (superheat and/or increased set point) is the fact that the tubes became as hot as 30°C during the flight test, which implies that the fluid temperature must have been even warmer. In the FLUINT model, the fluid was never warmer than 29°C because superheating was neglected and the reservoir was assumed to maintain a constant 29°C temperature. The most likely explanation is the lack of nonequilibrium compression of the vapor in the reservoir, which raises the set point temporarily (but perhaps long enough for this entire event). Better characterization of both the nonequilibrium reservoir response and the superheat in the evaporator would require more information and more detailed models. The data in this example was taken from the User's Manual for a computer program specifically designed to analyze CPLs. It is interesting to note that the predictions made by that program closely match those made by FLUINT, including falling short of the inlet pipe temperature by at least the same margin.

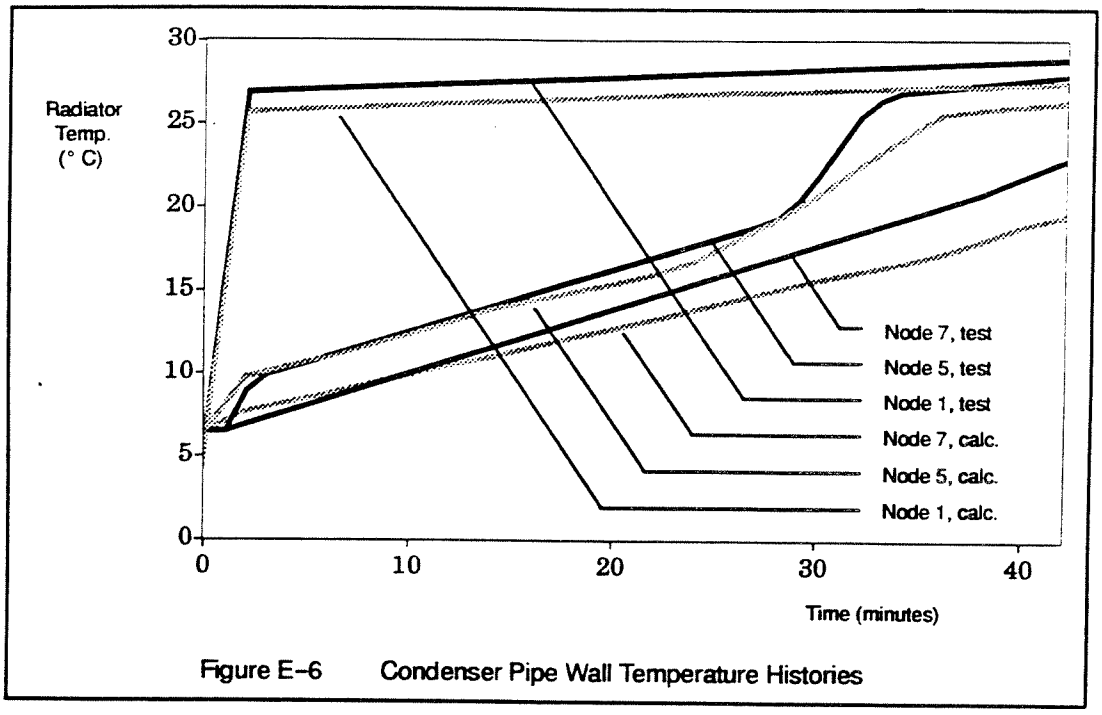


Figure E-6 Condenser Pipe Wall Temperature Histories

Examination of the output file reveals some warnings produced during the first run. A message is printed to notify the user that the isolator and the evaporator have deprimed because they are surrounded by liquid. These can be easily be ignored. The isolator never was primed and never should be. The evaporator is not primed until the liquid is cleared from one side, at which time a reprime advisory is printed. Note that a capillary device can still block vapor flow if deprimed unless the pressure difference exceeds the capillary limit. In the FLUINT sense, "primed" means either stopped flow (all vapor on one side and adiabatic) or a flowrate proportional to the heat input and *independent of pressure gradient.*

E.5 A Simplified Model for Comparison

As noted above, the real focus of this comparison is the condenser and radiator response. The above model can be tremendously simplified and still yield similar results if the start-up event is neglected. An input file for such a simplified analysis is provided after the input file for the detailed case.

Almost all details in the evaporator have been neglected. The evaporator pump is modeled simply as an MFRSET connector that pumps through a heater junction set for saturated vapor. Heater junctions are named by calls to HTRLMP. The flowrate is adjusted according to the difference in enthalpies between saturated vapor (as given by the enthalpy of the heater junction) and the enthalpy of the subcooled inlet. The inlet is represented by a tank containing the volume of the entire system. (Actually, since the vapor sides are swept clear fairly early in the transient, this volume should really be reduced to include only the liquid portions.) This model simply takes into account decreased subcooling in the inlet, which is a second-order correction needed to assure that the power input does not drift from 510W total. (The enthalpy in a heater junction is fixed and the QDOT required to maintain that state is calculated, not input.)

The thermal model of the evaporator section has been dropped, along with any detailed descriptions of flow losses. Because an MFRSET and heater junction representation of a capillary pump cannot deprime, such details are meaningless. The thermal model of the plate and the fluid model of the condenser are nearly exactly the same as above. Thus, most of the fluid model is really used to set the correct boundary conditions for the condenser.

An fast but limited-range description of ammonia was used to greatly reduce run time with virtually no loss of detail or accuracy. The FPROP block was produced using RAPPR and is exact at 29°C, the constant saturation condition in this example. The range is determined by the error tolerance input to RAPPR: in this case 20% for liquid properties and 10% for vapor properties. The limiting factor in the range limit is usually liquid specific heat. Note that such large error tolerances do not affect the results, which closely match predictions made using the full library ammonia description. This is due to the dominance of two-phase energy exchange compared to single-phase exchange—the heat of vaporization is exact, and large errors in the liquid specific heat are negligible.

The same time savings could have similarly applied to the full model discussed previously. This alternate description and methods for producing similar 6000 series descriptions are discussed in the main volume of the User's Manual. RAPPR may also be used to produce simplified 8000 (gas only) and 9000 (liquid only) descriptions of standard library fluids.

Unlike the full model described in previous sections, a steady-state analysis is performed before the transient in this abbreviated model. The purpose of this steady-state is to initialize the flowrate and its associated effects throughout the system. Otherwise, smaller time steps will result in the transient because of poor initial conditions. To preserve the correct initial temperatures of the thermal submodel, DRPMOD and ADDMOD are used to place the thermal submodel into a boundary state during the steady-state run.

As with the restarted run in the previous model, the main time step limit was caused by SINDA: the characteristic time of the condenser (CSGMIN) was very small compared to the run time. (FLUINT UA's are summed into the SINDA CSGMIN product, but are otherwise not treated like G's in transients.) Requiring the use of diffusion nodes for the pipe itself is therefore the main cause of the time step limit. Replacing these pipe nodes with arithmetic nodes (as is done in this example) and moving their mass to the adjacent plate nodes ***reduces the run time threefold with no difference in results.***

The results of this analysis are identical to those plotted in Figure E–6. *These results are nearly indistinguishable from results of the previous run.* However, the computational cost for this run was a small fraction of the cost of the full model, not counting the savings in engineering time. This should help illustrate the importance of avoiding unnecessary details with FLUINT. Frivolous details are much more easily tolerated in a thermal model, where the phenomena are simpler and the response times generally orders-of-magnitude larger. **If you don't need it, don't ask for it. If you don't know it, don't include it.**

Input File

```
HEADER OPTIONS DATA
TITLE FLUINT SAMPLE MODEL 5 - CPL GET AWAY SPECIAL (GAS) EXPERIMENT
C FIRST RUN ++++++++++++
C       RSO    = cplgas.rsi
C       OUTPUT = cplgas.out
C       USER1  = cplgas.usr
C SECOND RUN ++++++++++++
        RSI    = cplgas.rsi
        OUTPUT = cplgas2.out
        USER1  = cplgas2.usr
C
C EMERGENCY MAP FILE
        QMAP   = cplgas.map
C
HEADER CONTROL CONSTANTS, GLOBAL
        UID    = SI
        ABSZRO = -273.16
        TIMEND = 120.0                  $ START FIRST TWO MINUTES
        OUTPUT = 6.0                    $ OUTPUT EVERY 6 SEC. INIT.
        OUTPTF = 1.0E30                 $ FAKE OUTPUT FOR FLUINT
        DTMAXF = 10.0                   $ MAX 10 SEC TIME STEP
C
HEADER USER DATA,GLOBAL
C FIRST RUN ++++++++++++
C       NTEST  = 0                      $ RESTART FLAG: SET TO RECORD
C SECOND RUN ++++++++++++
        NTEST  = 121                    $ RESTART FLAG: SET TO RECORD
        TSAT   = 29.0                   $ SATURATION PRESSURE
        IFLAG  = 0                      $ FLAG FOR KICK START OF CAPPMP
C FOR EVAPORATION, USE 2500 W/M2-K AND OUTER WICK AREA
        UAEVAP = 2500.0*0.0889*0.02286*3.1416
```

```
HEADER FLOW DATA,CPL,FID=717                    $ START LOOP DESCRIPTION
C
C SET MODEL DEFAULTS
C SET PRESSURE FOR ALL LUMPS AT SATURATION FOR 29C
C
LU DEF,TL = 6.6,      PL! = 1.1323E6,       XL = 0.0
PA DEF, FR = 0.0                            $ SHUT DOWN INITIALLY
C
C 0.0153 SQ IN TOTAL GROOVE AREA
C FOR 1-PHASE GROOVED LINE, USE HYD DIA ASSUM, AF = TOTAL
C PERIMETER = 16*3*0.034 + 16*0.014" = 1.856" = 4.71E-2 m
C FLOW AREA = 16.0*6.17E-7 + 0.25*PI*4.699E-3**2 = 2.72E-5
C
       DH = 4.*2.72E-5/4.71E-2              $ DH FOR MOST PIPING
       AF = 2.72E-5
C
C DEFINE TANKS IN EVAPORATOR
C
LU TANK,10,    VOL = 54.1E-6          $ LIQUID HEADER (COLD)
LU TANK,20,    VOL = 25.4E-6          $ ONE EVAP PUMP (LIQUID)
               COMP = 0.01/1.13E6     $ GUESS
C
C ADD DUPED VAPOR GROOVES AND LINE OUT OF EVAP PUMP (DUPED DOWNSTR)
C
LU JUNC,30
C
C VAPOR GROOVES (45 TOTAL)
C
PA CONN,30,30,35,     DUP = 45.0
               DEV = STUBE
               TLEN = 0.0889/2.0      $ USE HALF LENGTH ASSUMPTION
               DH = 1.4E-3
               AF = 1.5E-6
               UPF = 1.0
LU JUNC,35
C
C LINE TO HEADER
C
PA CONN,35,35,40
               DUPJ = 2.0             $ DOWNSTREAM DUPED TWICE
               DEV  = STUBE
               TLEN = 0.079
               DH = 6.6E-3,   AF = -1.0
               UPF = 1.0
C
C TIE OUTLET LINE TO HEADER MASS
C
T HTN,35,35, EVAP.1500, 35
```

```
C
C THE FOLLOWING COMPLIANCE IS VERY LARGE BUT THIS DOESN'T
C MATTER SINCE THE RATIO OF THE CAPILLARY PRESSURE HEAD
C TO THE ABS PRESSURE IS SMALL.  MAX VOL CHANGE ABOUT 0.2%
C THIS ASSURES MINIMAL SURGE FOR TRANSITION TO TWO-PHASE
C TO PREVENT ARTIFICIAL DEPRIME
C
LU TANK,40,    VOL = 20.0E-6          $ VAPOR HEADER
               COMP = 1.0/1.13E6      $ MORE COMPLIANCE FOR TRANSITION
C
PA CONN,40,40,50,     DEV = STUBE,   UPF     = 1.0
               TLEN = 0.127,  DH = 0.0141,   AF = -1.0
C
C TIE HEADER TO MASS TO ADD THAT INERTIA.
C USE AREA FRACTION OF 2.0 TO RETURN TO FULL AREA
C THEN MULTIPLY BY FACTOR OF 2.0 FOR IMPINGEMENT AND
C MIXING EFFECTS IN OUTLET
C
T HTN,40,40,EVAP.2000,        40, 2.0*2.0
C
LU JUNC,50
C
C DEFINE CAPILLARY DEVICES IN EVAPORATORS
C (LIQUID LINES IGNORED AS NEGLIGIBLE COMPARED TO WICK DROPS
C
PA CONN,300,10,20,    DUPI = 2.0     $ UPSTREAM DUPED TWICE
               DEV = CAPIL
               RC = 1.65E-5
C HALF OF CFC FOR ONE HALF OF ISOLATOR
               CFC = 2.0*3.1416*0.1270*2.6E-13/0.650939/2.0
C
M CAPPMP,2,TIE,1000,1000,2000,20,30,1000,EVAP.1000,
               RC = 1.65E-5
               CFC = 2.0*3.1416*0.0889*2.6E-13/0.719744
               XVH = 0.95             $ 95% DRY IS ENOUGH TO PRIME
               XVL = 0.9              $ 90% DRY IS TOO WET TO PRIME
               UA = 0.0               $ OFF UNTIL WALL HOT ENOUGH
C
C DEFINE CONDENSER: 9 CENTERED SEGMENTS OF 1.57 M TOTAL LENGTH
C
M HX,1,C,1,1,1,PLATE.1,50,100,       NSEG = 9       $ VAPOR LINE
       TLENT = 1.57
       DHS = 4.*2.72E-5/4.71E-2
       AFS = 2.72E-5
       LU = JUNC,     PA = STUBE     $ LOW TEMPORAL RESOLUTION
C
C DEFINE CONDENSER OUTLET JUNCTION
C
```

```
LU JUNC,100
C
C ADD 15 CM LINE TO LIQUID HEADER
C
PA CONN,100,100,10,   DEV = STUBE,   TLEN = 0.15
C
C FINALLY, ADD RESERVOIR AND LINE BETWEEN IT AND LIQUID HEADER
C INITIALIZE PLENUM AS TWO PHASE BUT USE LIQUID SUCTION
C (EASY WAY TO GET SATURATED LIQUID)
C
LU PLEN,99,    TL = 29.0, XL = 0.1          $ SAT LIQUID
C              VOL = 2.8E-4                 $ IF WERE TANK
C
PA CONN,99,10,99, STAT = RLS                $ LIQ FROM PLEN SIDE
       DEV = STUBE,   TLEN = 0.70           $ 70 CM
       FK = 1.0                             $ ENTRANCE/EXIT LOSS
C
HEADER NODE DATA, PLATE
C
C BOUNDARY
       -300,6.6,0.0
C
C PIPE NODES
C
       GEN,1,9,1,6.6,85.9/9.0
C
C RADIATOR NODES
C
       11,6.6,230.3                 $ FIRST HALF PASS
       GEN,12,4,1,6.6,216.75        $ NEXT TWO PASS
       GEN,16,4,1,6.6,289.0         $ LENGTH TWO PASSES
C
HEADER NODE DATA, EVAP
C
C  WALL OF EVAPORATOR
       1000,6.6,137.7
C  WALL OF OUTLET LINE
       1500,6.6,-1.0
C  WALL OF HEADER/OUTLET (INCLUDE EXHAUST MASS)
       2000,6.6,15.0
C
```

```
HEADER CONDUCTOR DATA,EVAP
C
C CONDUCTANCE EVAP TO OUTLET LINE MASS
       1000,1000,1500,0.04
C
C CONDUCTANCE TO HEADER NODE MASS
C NOTE DUAL ONE-WAY CONDUCTORS MEAN THAN PLATE SEES TWO EVAPS.
C
       1500,-1500,2000,2.0*0.04      $ HEADER SEES 2 EVAP
       1501,1500,-2000,1.0*0.04      $ EVAP SEES 1 HEADER
       2000,2000,PLATE.1,0.1         $ COND SEES 1 HEADER
C
HEADER SOURCE DATA, EVAP
C
       1000,255.0                    $ 255W (510W TOTAL)
C
HEADER CONDUCTOR DATA, PLATE
C
C BOUNDARY TO PLATE THROUGH COTHERM
C 63% and 51% FIN EFFICIENCIES
C
       301,11,300,25.6*0.63
       GEN,302,4,1,12,1,300,0,24.4*0.63
       GEN,306,4,1,16,1,300,0,32.4*0.51
C
C RADIAL: TUBE WALL AND SADDLE WELD
C
       GEN,101,9,1,1,1,11,1,17.2
C
C AXIAL: ALONG FLOW DIRECTION
C
       423,12,13,0.85
       445,14,15,0.85
       467,16,17,0.85
       489,18,19,1.13
C
C SIDE: ACROSS FLOW DIRECTION
C
       212,11,12,5.28
       224,12,15,5.28
       235,13,14,5.28
       247,14,17,5.28
       256,15,16,5.28
       279,17,18,3.17
       268,16,19,3.17
C
```

```
HEADER FLOGIC 0, CPL
C
C ADD ACCELERATION FACTORS IN GROOVES
C
       AC30           = - 1.0/(DL30*AF30**2)
C
C ADD ACCELERATION FACTORS IN VAPOR HEADER
C
       AC40           = - 1.0/(DL40*AF40**2)
C
       IF(IFLAG .EQ. 0)THEN
              IF(EVAP.T1000 .GT. TSAT)THEN
C EMPTY OUTLET JUNCTION AND START EVAPORATION TO KICK START CAPPMP
                     IFLAG          = 1
                     UA1000         = UAEVAP
                     CALL SPRIME('CPL',20,30,RC1000,1)
              ELSE
C KEEP EVAP NODE FROM OVERSHOOTING SATURATION BY MORE THAN 1 DEGREE
                     EVAP.DTMPCA    = TSAT + 1.0 - EVAP.T1000
              ENDIF
       ELSE
              EVAP.DTMPCA     = 5.0
       ENDIF
C
C ALLOW SMALL TIME STEP DURING CLEARING PROCESS,
C SHOULD TAKE BIGGER STEPS AFTER RESTART
C
       IF(XL40 .GE. 1.0 .OR. XL40 .LE. 0.0)THEN
              DTMINF         = 1.0E-3
              IF(NTEST .NE. 0) DTMINF             = 1.0E-2
       ELSE
              DTMINF         = 1.0E-5
       ENDIF
C
HEADER FLOGIC 1,CPL
C
C THIS LOGIC PRINTS A FLOMAP AND OTHER OUTPUT IF THE
C EVAPORATOR PUMPS DRY-OUT OR DEPRIME
C FLOGIC 2 WILL SIGNAL STOP
C
       IF(UA1000*GK1000 .GT. 0.0) THEN
              CALL LMPTAB('ALL')
              CALL TIETAB('ALL')
              CALL PTHTAB('ALL')
              CALL FLOMAP('ALL',1)
       ENDIF
C
```

```
HEADER FLOGIC 2,CPL
C
C THIS LOGIC STOPS SIMULATION IF THE
C EVAPORATOR PUMPS DRY-OUT OR DEPRIME
C
       IF(UA1000*GK1000 .GT. 0.0)  TIMEND          = TIMEN
C
HEADER VARIABLES 1, PLATE
C
C UPDATE SINK TEMPERATURE LINEARLY
       T300           = 6.6 + (17.0 - 6.6)*TIMEM/(42.0*60.0)
C
HEADER OUTPUT CALLS,PLATE
       CALL LMPTAB('ALL')
       CALL TIETAB('ALL')
       CALL PTHTAB('ALL')
       CALL TPRINT('ALL')
       WRITE(NUSER1,10)TIMEN/60.0,T1,T2,T3,T4,T5,T6,T7
     .     ,CPL.PL30-CPL.PL20,CPL.FR40
10      FORMAT(1X,1PG11.4,4X,8G10.3,2G11.4)
C
```

```
HEADER OPERATIONS DATA
C
BUILD GAS,PLATE,EVAP
BUILDF GAS,CPL
C
DEFMOD CPL
C
C ADD LOSS FOR SUDDEN CONTRACTION AT INLET OF CONDENSER,
C AND 4 CONDENSER BENDS IN APPROXIMATE LOCATION
C
       FK1     = 0.37
       FK3     = 0.4
       FK5     = 0.4
       FK7     = 0.4
       FK9     = 0.4
C
C CHECK TO SEE IF THIS IS SECOND RUN (RESTART)
C
       IF(NTEST .NE. 0)THEN          $ IF RESTART AVAIL:
              CALL RESTAR(NTEST)     $ RESET NETWORK
              IFLAG  = 1             $ CAPPMP ALREADY GOING
              TIMEND = 42.0*60.0     $ FINISH 42 MINUTES
              PLATE.OUTPUT = 120.0   $ TWO MINUTE OUTPUT INTVL
              CPL.OUTPTF = 120.0     $ TWO MINUTE OUTPUT INTVL
       ELSE
              TSAT   = VTS(PL99-PATMOS,CPL.FI) + ABSZRO
       ENDIF
C
C WRITE TRANSIENT HEADER
C
       WRITE(NUSER1,10)(MTEST,MTEST=1,7)
10      FORMAT(/' CPL - RADIATOR TEMPERATURE RESPONSE TO STEP INPUT,',
     . ' PLUS SYSTEM PRESSURE DROP AND RADIATOR FLOWRATE'//
     .  '   TIME (MIN)',T15,7(4X,'REG.',I2),4X,'DELTA P',4X,'FLOWRATE'/)
C
C RUN TRANSIENT (FINISH FOR RESTART RUN)
C
       CALL FWDBCK
       IF(NTEST .EQ. 0)CALL RESAVE('ALL')
C
```

```
HEADER SUBROUTINE DATA
DEBOFF
FSTART
C
C USE MODIFIED ROHSENOW CORRELATION: IF AF IS EQUAL TO CORRECT
C VALUE AND XL IS GREATER THAN 0.025, THEN USE KAMOTANI'S
C AVERAGED WITH ROHSEN ELSE BLEND INTO ROHSENOW AT LOW QUALITY
C
      FUNCTION ROHSEN(WA,D,AF,TW,P,T,X,ALF,FI)
C
C Purpose:
C      ROSHENHOW'S CONDENSATION HEAT TRANSFER CORRELATION
C      (BASED ON WORK BY TRAVISS)
C
C
C Argument List Definitions:
C      WA     - Abs value of flowrate
C      D      - Diameter
C      AF     - Path flow area
C      TW     - Wall (node) temperature
C      P      - Total gas pressure
C      T      - Temperature
C      X      - Quality
C      ALF    - Void fraction
C      FI     - Fluid identifier array
C
C---------END OF DESCRIPTION -------
C
      DOUBLE PRECISION P
      INTEGER FI(1)
      COMMON /LSTSAT/ DLIQ,DVAP,HFG
C
C      FX     XTT
C      0.1    24.203035
C      0.15   11.023539
C      0.5    1.2684654
C      1.0    0.42202932
C      20.0   9.347339E-3
C
      SAVE XTTCO,CUTOFF
      SAVE QUALOW,AFG,DIAVC
      DATA QUALOW/0.025/,AFG/2.72E-5/,DIAVC/4.7E-3/
      DATA XTTCO,CUTOFF/1.2684654,0.50/
C
      IF(FI(2) .EQ. 8 .OR. FI(2) .EQ. 9)THEN
            CALL ABNORM('ROHSEN',FI(1),' ILLEGAL FLUID ')
            RETURN
      ENDIF
```

```
      TLIQ          = 0.75*TW + 0.25*T
C
C BLEND IN WITH SINGLE PHASE
C
      IF(ALF .LT. 0.5)THEN
            FRACT         = 1.5*ALF
            TLIQ          = FRACT*TW + (1.0-FRACT)*T
      ENDIF
      TKL           = VCONDF(TLIQ,FI)
C
      VISL          = VVISCF(TLIQ,FI)
      VISV          = VVISCV(P,T,FI)
      IF(DLIQ .NE. 0.0)THEN
            RHOL          = DLIQ
            RHOV          = DVAP
      ELSE
            RHOL          = VDL(TLIQ,FI)
            RHOV          = 1.0/VSV(P,T,FI)
      ENDIF
C
C THE MARTINELLI PARAMETER WAS LISTED IN 1973 ROHSENOW/HARNETT
C HANDBOOK OF HEAT TRANSFER AS:
      XTTTMP        = (VISL/VISV)**(0.1)*SQRT(RHOV/RHOL)
C
C THIS IS THE SAME AS THAT USED IN CHEN'S EVAPORATIVE CORRELATION
C AND MANY OTHER SOURCES.  NOW, BOTH THE 1985 VERSION AND
C HESTRONI'S HANDBOOK OF MULTIPHASE SYSTEMS LIST THE VISCOSITY
C RATIO INVERTED:
C     XTTTMP        = (VISV/VISL)**(0.1)*SQRT(RHOV/RHOL)
C THE ABOVE DISAGREES WITH THE ORIGINAL PAPER (TRAVISS 1973)
C AND WILL BE IGNORED.  COEFF. IS UP TO 80% LOWER AS A RESULT.
C
      XTT           = XTTTMP*((1.0-X)/X)**(0.9)
      FX            = 0.15*( 1.0/XTT + 2.85*XTT**(-0.476) )
      CPL           = VCPF(P,TLIQ,FI)
      PRL           = VISL*CPL/TKL
C
C OFFICIALLY, BREAKPOINTS ARE 50 AND 1125.  USE FOLLOWING FOR LESS
C DISCONTINUITY IN REYNOLDS NUMBER
C
      IF(FX .LT. CUTOFF) THEN
            Y             = (XTTCO/XTTTMP)**(1.0/0.9)
            XC            = 1.0/(1.0 + Y)
            REL           = WA*(1.0-XC)*D/(AF*VISL)
      ELSE
            REL           = WA*(1.0-X)*D/(AF*VISL)
      ENDIF
C
```

```
      IF(REL .GE. 1200.0)THEN
            F2              = 5.0*PRL + 5.0*ALOG(1.0+5.0*PRL)
     .                      + 2.5*ALOG(0.0031*REL**(0.812))
      ELSE IF( REL .GE. 52.3 )THEN
            F2              = 5.0*PRL + 5.0*ALOG(1.0 + PRL*
     .                      (0.09636*REL**(0.585)-1.0))
      ELSE
            F2              = 0.707*PRL*SQRT(REL)
      ENDIF
C
      HL              = DITTUS(WA,D,AF,TW,P,T,0.0,FI)
C
      IF( FX .GE. 1.0) THEN
C
C     SKIP TRANSITION TO SINGLE-PHASE VAPOR (INHERENTLY STABLE)
C     JUST AS TRANSITION TO SINGLE-PHASE LIQUID IS STABLE FOR BOILING
C
            XNU             = FX**(1.15)*PRL*REL**(0.9)/F2
            HCOND           = XNU*TKL/D
      ELSEIF( FX .GE. CUTOFF )THEN
            XNU             = FX*PRL*REL**(0.9)/F2
            HCOND           = XNU*TKL/D
      ELSE
C
C LOW QUALITY, INTERPOLATE WITH LIQUID DITTUS
C MAKE SURE TWO PHASE H IS >= SINGLE PHASE H AT SAME FLOWRATE
C (THIS IS NEARLY THE SAME AS THE SLUG FLOW TRANSITION
C HCOND SCALES ROUGHLY WITH X**.8 FOR LOW X. ( FX > 1.0)
C HCOND SCALES ROUGHLY WITH X**.7 FOR LOW X. ( FX < 1.0)
C (THIS IS SIMILAR TO X**.76 IN THE SHAH CORRELATION)
C SCALE POWER WITH QUALITY TO LESSEN XL=0 DISCONTINUITY
C
            XNU             = CUTOFF*PRL*REL**(0.9)/F2
            HCOND           = XNU*TKL/D
            IF(HCOND .GT. HL)THEN
                  RATIO             = X/XC
                  XPOW              = 1.0 - 0.3*RATIO
                  HCOND             = HL + (HCOND-HL)*RATIO**XPOW
            ENDIF
      ENDIF
C
      R               = MAX(HL,HCOND)
C
C CORRECTED KAMOTANI CORRELATION FOR HEAT TRANSFER
C USING HEAT PIPE TEST DATA
C
      IF((ABS(AF/AFG) - 1.0) .LE. 1.0E-5)THEN
            HG              = 1.083E3*TKL/
```

```
                              (0.0221+TKL*.034/(.014*165.))
C
C TAKE RATIO OF HEAT TRANSFER AREA TO OLD WETTED PERIMETER
C USE VAPOR CORE DIAMETER AS BASIS FOR KAMOTANI'S CORRELATION
C
              HG              = HG*3.1416*DIAVC/(4.*AF/D)
C
C SMOOTH IN GROOVE RESULTS IF ACTIVE WITH CUBIC SMOOTHING FUNCTION
C
              IF(X .LT. QUALOW)THEN
                     RATIO           = X/QUALOW
                     SMOOTH          = (3. - 2.*RATIO)*RATIO**2
                     HG              = HG*SMOOTH + (1.0-SMOOTH)*R
              ENDIF
              R               = 0.5*(HG+R)
       ENDIF
C
       ROHSEN          = R
C
       RETURN
       END
END OF DATA
```

```
C ===================================================================
C THIS IS THE INPUT FILE FOR THE SIMPLIFIED CPL START-UP, FOCUSING
C ON THE THERMAL EVENT AND NEGLECTING THE SHORT HYDRODYNAMIC EVENT
C ===================================================================
C
HEADER OPTIONS DATA
TITLE FLUINT SAMPLE MODEL 6 - SIMPLIFIED CPL GET AWAY SPECIAL (GAS)
EXPERIMENT
       OUTPUT = cplpdq.out
       USER1  = cplpdq.usr
C
HEADER CONTROL CONSTANTS, GLOBAL
       UID    = SI
       ABSZRO = -273.16
       OUTPUT = 120.0              $ OUTPUT EVERY 2 MINUTES
       OUTPTF = 120.0
       DTMAXF = 10.0               $ MAX 10 SEC TIME STEP
       NLOOPS = 200
C
HEADER USER DATA, GLOBAL
       POWER  = 2.0*255.0          $ 510.0 WATTS TOTAL
C
C GET FAST FLUID DESC. PRODUCED BY RAPPR FOR NH3 @ 29C
C
INCLUDE cplpdq.inc
C
HEADER FLOW DATA,CPL,FID=6717      $ START LOOP DESCRIPTION
C
C SET MODEL DEFAULTS
C SET PRESSURE FOR ALL LUMPS AT SATURATION FOR 29C
C
LU DEF,TL = 6.6,      PL! = 1.1323E6,       XL = 0.0
PA DEF,FR = 0.0                    $ SHUT DOWN INITIALLY
C
C 0.0153 SQ IN TOTAL GROOVE AREA
C FOR 1-PHASE GROOVED LINE, USE HYD DIA ASSUM, AF = TOTAL
C PERIMETER = 16*3*0.034 + 16*0.014" = 1.856" = 4.71E-2 m
C FLOW AREA = 16.0*6.17E-7 + 0.25*PI*4.699E-3**2 = 2.72E-5
C
              DH = 4.*2.72E-5/4.71E-2
              AF = 2.72E-5
C
C DEFINE SINGLE SUBCOOLED TANK AS EVAPORATOR INLET
C
LU TANK,10,    VOL = 125.0E-6      $ LIQUID HEADER (COLD)
               COMP = 0.01/1.13E6  $ GUESS
C
C USE MFRSET AS PUMP
```

```
C
PA CONN,1040,10,40,   DEV = MFRSET
C
C CONDENSER INLET (HEATER JUNCTION)
C SET AS SATURATED VAPOR
C
LU JUNC,40,    XL = 1.0       $ SAT VAPOR
C
C DEFINE CONDENSER: 9 CENTERED SEGMENTS OF 1.57 M TOTAL LENGTH
C WILL USE MODIFIED ROHSENOW CORRELATION FOR GROOVE HEAT TRANSFER
C IF APPROPROPRIATE AF IS USED
C
M HX,1,C,1,1,1,PLATE.1,40,10,        NSEG = 9       $ VAPOR LINE
       TLENT = 1.57
       DHS = 4.*2.72E-5/4.71E-2
       AFS = 2.72E-5
       LU = JUNC,     PA = STUBE     $ LOW TEMPORAL RESOLUTION
C
C FINALLY, ADD RESERVOIR AND LINE BETWEEN IT AND LIQUID HEADER
C INITIALIZE PLENUM AS TWO PHASE BUT USE LIQUID SUCTION
C (EASY WAY TO GET SATURATED LIQUID)
C
LU PLEN,99,    XL = 0.1                    $ SAT LIQUID
C              VOL = 2.8E-4                $ IF WERE TANK
C
PA CONN,99,10,99, STAT = RLS               $ LIQ FROM PLEN SIDE
               DEV = STUBE,   TLEN = 0.70  $ 70 CM
               FK = 1.0                    $ ENTRANCE LOSS
C
HEADER NODE DATA, PLATE
C
C BOUNDARY
       -300,6.6,0.0
C
C PIPE NODES: MOVE MASS INTO HEAVIER RADIATOR NODES
C
C      GEN,1,9,1,6.6,85.9/9.0
       GEN,1,9,1,6.6,-1.0
C
C RADIATOR NODES
C
       11,6.6, 85.9/9.0 + 230.3            $ FIRST HALF PASS
       GEN,12,4,1,6.6, 85.9/9. + 216.75    $ NEXT TWO PASS
       GEN,16,4,1,6.6, 85.9/9.0 + 289.0    $ LENGTH TWO PASSES
C
```

```
HEADER CONDUCTOR DATA, PLATE
C
C BOUNDARY TO PLATE THROUGH COTHERM
C 63% and 51% FIN EFFICIENCIES
C
       301,11,300,25.6*0.63
       GEN,302,4,1,12,1,300,0,24.4*0.63
       GEN,306,4,1,16,1,300,0,32.4*0.51
C
C RADIAL: TUBE WALL AND SADDLE WELD
C
       GEN,101,9,1,1,1,11,1,17.2
C
C AXIAL: ALONG FLOW DIRECTION
C
       423,12,13,0.85
       445,14,15,0.85
       467,16,17,0.85
       489,18,19,1.13
C
C SIDE: ACROSS FLOW DIRECTION
C
       212,11,12,5.28
       224,12,15,5.28
       235,13,14,5.28
       247,14,17,5.28
       256,15,16,5.28
       279,17,18,3.17
       268,16,19,3.17
C
HEADER FLOGIC 0, CPL
C
C MAKE SURE HEATER JUNCTION STAYS VAPOR
C
       CALL CHGLMP('CPL',40,'XL',1.0,'PL')
C
C SET MFRSET FOR CORRECT VOLUMETRIC FLOWRATE
C
       SMFR1040       = POWER/(HL40 - HL10)
C
HEADER VARIABLES 1, PLATE
C
C UPDATE SINK TEMPERATURE LINEARLY
       T300           = 6.6 + (17.0 - 6.6)*TIMEM/(42.0*60.0)
C
```

```
HEADER OUTPUT CALLS,CPL
       CALL LMPTAB('ALL')
       CALL TIETAB('ALL')
       CALL PTHTAB('ALL')
HEADER OUTPUT CALLS,PLATE
       CALL TPRINT('ALL')
       IF(NSOL .GT. 1)WRITE(NUSER1,10)TIMEN/60.0,T1,T2,T3,T4,T5,T6,T7
     .     ,CPL.PL40-CPL.PL10,CPL.FR1040
10      FORMAT(1X,1PG11.4,4X,8G10.3,2G11.4)
C
HEADER OPERATIONS DATA
C
BUILD GAS,PLATE
BUILDF GAS,CPL
C
DEFMOD CPL
C
C HOLD HEATER JUNCTION ENTHALPY
C
       CALL HTRLMP('CPL',40)
C
C RUN STEADY STATE TO GET FLUID INITIAL CONDITIONS HOLDING THERMAL
C HOLD TANK 10 TOO SINCE INITIALLY NO FLOW
C
       CALL DRPMOD('PLATE')
       CALL HLDLMP('CPL',10)
       CALL FASTIC
       CALL ADDMOD('PLATE')
       CALL RELLMP('CPL',10)
C
C RUN TRANSIENT
C
C WRITE TRANSIENT HEADER
C
       WRITE(NUSER1,10)(MTEST,MTEST=1,7)
10      FORMAT(/' CPL - RADIATOR TEMPERATURE RESPONSE TO STEP INPUT,',
     . ' PLUS SYSTEM PRESSURE DROP AND RADIATOR FLOWRATE'//
     .  '    TIME (MIN)',T15,7(4X,'REG.',I2),4X,'DELTA P',4X,'FLOWRATE'/)
C
       TIMEND = 42.0*60.0          $ START 42 MINUTES TRANSIENT
       CALL FWDBCK
C
```

```
HEADER SUBROUTINE DATA
DEBOFF
FSTART
C
C USE MODIFIED ROHSENOW CORRELATION: IF AF IS EQUAL TO CORRECT
C VALUE AND XL IS GREATER THAN 0.025, THEN USE KAMOTANI'S
C AVERAGED WITH ROHSEN ELSE BLEND INTO ROHSENOW AT LOW QUALITY
C
      FUNCTION ROHSEN(WA,D,AF,TW,P,T,X,ALF,FI)
C
C Purpose:
C      ROSHENHOW'S CONDENSATION HEAT TRANSFER CORRELATION
C      (BASED ON WORK BY TRAVISS)
C
C
C Argument List Definitions:
C      WA     - Abs value of flowrate
C      D      - Diameter
C      AF     - Path flow area
C      TW     - Wall (node) temperature
C      P      - Total gas pressure
C      T      - Temperature
C      X      - Quality
C      ALF    - Void fraction
C      FI     - Fluid identifier array
C
C---------END OF DESCRIPTION -------
C
      DOUBLE PRECISION P
      INTEGER FI(1)
      COMMON /LSTSAT/ DLIQ,DVAP,HFG
C
C      FX     XTT
C      0.1    24.203035
C      0.15   11.023539
C      0.5    1.2684654
C      1.0    0.42202932
C      20.0   9.347339E-3
C
      SAVE XTTCO,CUTOFF
      SAVE QUALOW,AFG,DIAVC
      DATA QUALOW/0.025/,AFG/2.72E-5/,DIAVC/4.7E-3/
      DATA XTTCO,CUTOFF/1.2684654,0.50/
C
      IF(FI(2) .EQ. 8 .OR. FI(2) .EQ. 9)THEN
            CALL ABNORM('ROHSEN',FI(1),' ILLEGAL FLUID ')
            RETURN
      ENDIF
```

```
      TLIQ          = 0.75*TW + 0.25*T
C
C BLEND IN WITH SINGLE PHASE
C
      IF(ALF .LT. 0.5)THEN
            FRACT         = 1.5*ALF
            TLIQ          = FRACT*TW + (1.0-FRACT)*T
      ENDIF
      TKL           = VCONDF(TLIQ,FI)
C
      VISL          = VVISCF(TLIQ,FI)
      VISV          = VVISCV(P,T,FI)
      IF(DLIQ .NE. 0.0)THEN
            RHOL          = DLIQ
            RHOV          = DVAP
      ELSE
            RHOL          = VDL(TLIQ,FI)
            RHOV          = 1.0/VSV(P,T,FI)
      ENDIF
C
C THE MARTINELLI PARAMETER WAS LISTED IN 1973 ROHSENOW/HARNETT
C HANDBOOK OF HEAT TRANSFER AS:
      XTTTMP        = (VISL/VISV)**(0.1)*SQRT(RHOV/RHOL)
C
C THIS IS THE SAME AS THAT USED IN CHEN'S EVAPORATIVE CORRELATION
C AND MANY OTHER SOURCES.  NOW, BOTH THE 1985 VERSION AND
C HESTRONI'S HANDBOOK OF MULTIPHASE SYSTEMS LIST THE VISCOSITY
C RATIO INVERTED:
C     XTTTMP        = (VISV/VISL)**(0.1)*SQRT(RHOV/RHOL)
C THE ABOVE DISAGREES WITH THE ORIGINAL PAPER (TRAVISS 1973)
C AND WILL BE IGNORED.  COEFF. IS UP TO 80% LOWER AS A RESULT.
C
      XTT           = XTTTMP*((1.0-X)/X)**(0.9)
      FX            = 0.15*( 1.0/XTT + 2.85*XTT**(-0.476) )
      CPL           = VCPF(P,TLIQ,FI)
      PRL           = VISL*CPL/TKL
C
C OFFICIALLY, BREAKPOINTS ARE 50 AND 1125.  USE FOLLOWING FOR LESS
C DISCONTINUITY IN REYNOLDS NUMBER
C
      IF(FX .LT. CUTOFF) THEN
            Y             = (XTTCO/XTTTMP)**(1.0/0.9)
            XC            = 1.0/(1.0 + Y)
            REL           = WA*(1.0-XC)*D/(AF*VISL)
      ELSE
            REL           = WA*(1.0-X)*D/(AF*VISL)
      ENDIF
C
```

```
        IF(REL .GE. 1200.0)THEN
               F2              = 5.0*PRL + 5.0*ALOG(1.0+5.0*PRL)
     .                         + 2.5*ALOG(0.0031*REL**(0.812))
        ELSE IF( REL .GE. 52.3 )THEN
               F2              = 5.0*PRL + 5.0*ALOG(1.0 + PRL*
     .                         (0.09636*REL**(0.585)-1.0))
        ELSE
               F2              = 0.707*PRL*SQRT(REL)
        ENDIF
C
        HL             = DITTUS(WA,D,AF,TW,P,T,0.0,FI)
C
        IF( FX .GE. 1.0) THEN
C
C       SKIP TRANSITION TO SINGLE-PHASE VAPOR (INHERENTLY STABLE)
C       JUST AS TRANSITION TO SINGLE-PHASE LIQUID IS STABLE FOR BOILING
C
               XNU             = FX**(1.15)*PRL*REL**(0.9)/F2
               HCOND           = XNU*TKL/D
        ELSEIF( FX .GE. CUTOFF )THEN
               XNU             = FX*PRL*REL**(0.9)/F2
               HCOND           = XNU*TKL/D
        ELSE
C
C LOW QUALITY, INTERPOLATE WITH LIQUID DITTUS
C MAKE SURE TWO PHASE H IS >= SINGLE PHASE H AT SAME FLOWRATE
C (THIS IS NEARLY THE SAME AS THE SLUG FLOW TRANSITION
C HCOND SCALES ROUGHLY WITH X**.8 FOR LOW X. ( FX > 1.0)
C HCOND SCALES ROUGHLY WITH X**.7 FOR LOW X. ( FX < 1.0)
C (THIS IS SIMILAR TO X**.76 IN THE SHAH CORRELATION)
C SCALE POWER WITH QUALITY TO LESSEN XL=0 DISCONTINUITY
C
               XNU             = CUTOFF*PRL*REL**(0.9)/F2
               HCOND           = XNU*TKL/D
               IF(HCOND .GT. HL)THEN
                      RATIO          = X/XC
                      XPOW           = 1.0 - 0.3*RATIO
                      HCOND          = HL + (HCOND-HL)*RATIO**XPOW
               ENDIF
        ENDIF
C
        R              = MAX(HL,HCOND)
C
C CORRECTED KAMOTANI CORRELATION FOR HEAT TRANSFER
C USING HEAT PIPE TEST DATA
C
        IF((ABS(AF/AFG) - 1.0) .LE. 1.0E-5)THEN
               HG              = 1.083E3*TKL/
```

```
     .                          (0.0221+TKL*.034/(.014*165.))
C
C TAKE RATIO OF HEAT TRANSFER AREA TO OLD WETTED PERIMETER
C USE VAPOR CORE DIAMETER AS BASIS FOR KAMOTANI'S CORRELATION
C
               HG             = HG*3.1416*DIAVC/(4.*AF/D)
C
C SMOOTH IN GROOVE RESULTS IF ACTIVE WITH CUBIC SMOOTHING FUNCTION
C
               IF(X .LT. QUALOW)THEN
                      RATIO          = X/QUALOW
                      SMOOTH         = (3. - 2.*RATIO)*RATIO**2
                      HG             = HG*SMOOTH + (1.0-SMOOTH)*R
               ENDIF
               R              = 0.5*(HG+R)
       ENDIF
C
       ROHSEN         = R
C
       RETURN
       END
END OF DATA
```

```
C ===================================================================
C RAPPR GENERATED INCLUDE FILE FOR ALTERNATE INPUT FILE
C ===================================================================
C
HEADER FPROP DATA, 6717, SI , 0.0
C
C   THIS FLUID DESCRIPTION WAS CREATED BY RAPPR FOR FLUID  717
C   AND IS ACCURATE FOR TEMPERATURES NEAR   302.15
C   MAXIMUM ERROR IN LIQUID PROPERTIES = 20.00 PERCENT
C   MAXIMUM ERROR IN VAPOR PROPERTIES  = 10.00 PERCENT
C   MAXIMUM LIQUID (SATURATION) TEMP. =   331.55490
C
         TCRIT =     405.59998    ,    PCRIT =     11282378.
         TGMAX =     331.55490    ,    PGMAX =     2496161.5
         TMIN =      267.78616
C
VINIT
       DOUBLE PRECISION POLD,PSAT
       SAVE TOLD,TTOC,A,B,BINV,C,CINV,TREF,DLIQ,DLDT
       SAVE CPVAP,CPVDT,REFF,BC,POLD,TTOCX,TSAT,DLDP,DLDPT
       DATA A,B,BINV/  20.802042    , -12773.754    ,-7.82855204E-05/
       DATA C,CINV,TREF/ -1.3183613    ,-0.75851744    ,  302.14999    /
       DATA DLIQ,DLDT/  596.95667    , -1.5352054    /
       DATA CPVAP,CPVDT,TOLD,TSAT/  3058.6499    ,  24.452654    ,2*0.0/
       DATA REFF,BC,POLD/  426.70825    ,  16840.422    ,0.0D0/
       DATA DLDP,DLDPT/ 9.64974333E-07, 1.29314524E-08/
C
VSV
       V      = REFF*T/SNGL(P)
C
VDL
       D      = DLIQ + (T-TREF)*DLDT
C
VCPV
       CP     = CPVAP + (T-TREF)*CPVDT
C
VTAV2
       T      = TLOOKS(H,CPVAP,CPVDT,TREF,0.0,0.0)
       V      = REFF*T/SNGL(P)
C
VH
       H      = HINTS(CPVAP,CPVDT,TREF,0.0,T)
C
VTS
       IF(P .NE. POLD)THEN
            TTOCX  = (ALOG(SNGL(P))-A)*BINV
            TSAT   = TTOCX**CINV
            POLD   = P
```

```
      ENDIF
      T       = TSAT
C
VDPDT
      IF(P .NE. POLD)THEN
           TTOCX  = (ALOG(SNGL(P))-A)*BINV
           TSAT   = TTOCX**CINV
           POLD   = P
      ENDIF
      DPDT    = BC*SNGL(P)*TTOCX/TSAT
C
VHFG
      IF(T .NE. TOLD)THEN
           IF(T .EQ. TSAT)THEN
                TTOC   = TTOCX
                TOLD   = TSAT
           ELSE
                TTOC   = T**C
                TOLD   = T
           ENDIF
      ENDIF
      VLIQ    = 1.0/(DLIQ + (T-TREF)*DLDT)
      HFG     = BC*SNGL(P)*(V-VLIQ)*TTOC
C
VPS
      IF(T .NE. TOLD)THEN
           IF(T .EQ. TSAT)THEN
                TTOC   = TTOCX
                TOLD   = TSAT
           ELSE
                TTOC   = T**C
                TOLD   = T
           ENDIF
      ENDIF
      P       = DBLE(EXP(A+B*TTOC))
C
VDPDTT
      IF(T .NE. TOLD)THEN
           IF(T .EQ. TSAT)THEN
                TTOC   = TTOCX
                TOLD   = TSAT
           ELSE
                TTOC   = T**C
                TOLD   = T
           ENDIF
      ENDIF
      DPDT    = BC*EXP(A+B*TTOC)*TTOC/T
C
```

```
VCONDF
       COND    =  0.47210324      + (T-TREF)*(-2.31996831E-03)
C
VVISCF
       VISC    =  1.37435200E-04 + (T-TREF)*(-1.49908169E-06)
C
VCONDV
       COND    =  2.48712618E-02 + (T-TREF)*( 1.05783671E-04)
C
VVISCV
       VISC    =  1.03307348E-05 + (T-TREF)*( 3.60964201E-08)
C
VST
       ST      =  1.91859473E-02 + (T-TREF)*(-2.33207902E-04)
C
VDLC
       IF(T .NE. TOLD)THEN
            IF(T .EQ. TSAT)THEN
                 TTOC    = TTOCX
                 TOLD    = TSAT
            ELSE
                 TTOC    = T**C
                 TOLD    = T
            ENDIF
       ENDIF
       PSAT    = DBLE(EXP(A+B*TTOC))
       DSAT    =   596.95667      + (T-TREF)*( -1.5352054     )
       DLD     =  9.64974333E-07 + (T-TREF)*( 1.29314524E-08)
       D       = DSAT + MAX(0.0,SNGL(P-PSAT))*DLD
C
VS
       PHI     = SINTS(CPVAP,CPVDT,TREF,   267.78616     ,T)
       S       = PHI - REFF*ALOG(REFF*T/(V*  349573.94      ))
C
C ================================================================
C END OF ALTERNATE INPUTS
C ================================================================
```

Processor Output

CPL - RADIATOR TEMPERATURE RESPONSE TO STEP INPUT, PLUS SYSTEM PRESSURE DROP AND RADIATOR FLOWRATE

TIME (MIN)	REG. 1	REG. 2	REG. 3	REG. 4	REG. 5	REG. 6	REG. 7	DELTA P	FLOWRATE
0.	6.60	6.60	6.60	6.60	6.60	6.60	6.60	0.	0.
0.1000	6.60	6.60	6.60	6.60	6.60	6.60	6.60	2.885E-03	6.3443E-08
0.2000	6.62	6.61	6.61	6.61	6.61	6.61	6.61	8.793E-03	2.0068E-07
0.3000	7.35	6.78	6.66	6.64	6.63	6.63	6.63	212.	3.1301E-04
0.4000	22.4	13.8	10.0	8.42	7.71	7.24	7.04	314.	5.0233E-04
0.5000	24.2	21.9	9.76	7.81	7.40	6.94	6.87	353.	3.5488E-04
0.6000	25.0	23.4	13.2	8.81	7.97	7.05	6.90	383.	3.7923E-04
0.7000	25.4	24.0	16.2	9.64	8.52	7.19	6.98	400.	3.9416E-04
0.8000	25.6	24.3	18.4	10.1	8.87	7.32	7.07	408.	4.0042E-04
0.9000	25.7	24.4	19.9	10.5	9.13	7.44	7.17	412.	4.0352E-04
1.000	25.7	24.5	20.9	10.8	9.31	7.54	7.26	414.	4.0512E-04
1.100	25.8	24.6	21.5	11.0	9.44	7.62	7.35	416.	4.0587E-04
1.200	25.8	24.6	22.0	11.2	9.53	7.69	7.42	416.	4.0630E-04
1.300	25.8	24.6	22.2	11.4	9.61	7.75	7.49	417.	4.0654E-04
1.400	25.8	24.6	22.4	11.5	9.67	7.79	7.55	417.	4.0663E-04
1.500	25.8	24.6	22.6	11.5	9.71	7.84	7.60	417.	4.0671E-04
1.600	25.8	24.7	22.7	11.6	9.76	7.87	7.64	418.	4.0675E-04
1.700	25.8	24.7	22.8	11.7	9.79	7.91	7.68	418.	4.0678E-04
1.800	25.8	24.7	22.9	11.7	9.82	7.94	7.72	418.	4.0680E-04
1.900	25.8	24.7	22.9	11.8	9.85	7.97	7.75	418.	4.0682E-04
2.000	25.8	24.7	23.0	11.8	9.88	8.00	7.78	418.	4.0684E-04
2.000	25.8	24.7	23.0	11.8	9.88	8.00	7.78	418.	4.0680E-04

CPL - RADIATOR TEMPERATURE RESPONSE TO STEP INPUT, PLUS SYSTEM PRESSURE DROP AND RADIATOR FLOWRATE

TIME (MIN)	REG. 1	REG. 2	REG. 3	REG. 4	REG. 5	REG. 6	REG. 7	DELTA P	FLOWRATE
2.000	25.8	24.7	23.0	11.8	9.88	8.00	7.78	418.	4.0680E-04
4.000	25.9	24.8	23.7	12.6	10.5	8.53	8.32	419.	4.0714E-04
6.000	26.0	25.0	24.0	13.7	11.1	9.10	8.88	421.	4.0759E-04
8.000	26.1	25.1	24.2	14.8	11.8	9.68	9.44	422.	4.0837E-04
10.00	26.2	25.3	24.4	15.9	12.5	10.3	10.0	422.	4.0912E-04
12.00	26.3	25.4	24.7	17.0	13.2	10.9	10.6	423.	4.0989E-04
14.00	26.4	25.5	24.9	18.2	14.0	11.4	11.1	424.	4.1068E-04
16.00	26.5	25.7	25.1	19.6	14.5	12.0	11.7	425.	4.1145E-04
18.00	26.6	25.8	25.3	21.0	15.0	12.5	12.3	426.	4.1223E-04
20.00	26.7	25.9	25.5	22.4	15.5	13.0	12.8	427.	4.1304E-04
22.00	26.8	26.0	25.6	23.8	16.1	13.5	13.4	427.	4.1383E-04
24.00	26.9	26.2	25.8	24.9	16.9	14.1	14.0	429.	4.1463E-04
26.00	27.0	26.3	26.0	25.1	18.2	15.0	14.6	432.	4.1544E-04
28.00	27.0	26.4	26.1	25.4	19.5	15.8	15.2	434.	4.1628E-04
30.00	27.1	26.6	26.2	25.6	20.9	16.7	15.8	435.	4.1713E-04
32.00	27.2	26.7	26.4	25.8	22.6	17.3	16.3	438.	4.1800E-04
34.00	27.3	26.8	26.5	26.0	24.2	18.0	16.8	439.	4.1884E-04
36.00	27.4	27.0	26.7	26.1	25.7	18.9	17.4	442.	4.1969E-04
38.00	27.4	27.1	26.8	26.3	26.0	20.4	18.2	446.	4.2059E-04
40.00	27.5	27.2	26.9	26.5	26.2	22.0	19.1	449.	4.2159E-04
42.00	27.6	27.3	27.0	26.6	26.4	23.9	19.7	453.	4.2262E-04

FLUID SUBMODEL NAME = CPL ; FLUID NO. = 717

MAX TIME STEP = 0. ; LIMITING MESSAGE = (NO SOLUTION STEP TAKEN YET)
LAST TIME STEP = 0. VS. DTMAXF/DTMINF = 10.0000 / 1.000000E-03; AVERAGE TIME STEP = 0.
PROBLEM TIME TIMEN = 0. VS. TIMEND = 120.000

LUMP PARAMETER TABULATION FOR SUBMODEL CPL

LUMP	TYPE	TEMP	PRESSURE	QUALITY	DENSITY	ENTHALPY	HEAT RATE	MASS RATE	ENERGY RATE
10	TANK	6.600	1.1323E+06	0.	629.6	2.1271E+05	0.	0.	0.
20	TANK	6.600	1.1323E+06	0.	629.6	2.1271E+05	0.	0.	0.
40	TANK	6.600	1.1323E+06	0.	629.6	2.1271E+05	0.	0.	0.
30	JUNC	6.600	1.1323E+06	0.	629.6	2.1271E+05	0.	0.	0.
35	JUNC	6.600	1.1323E+06	0.	629.6	2.1271E+05	0.	0.	0.
50	JUNC	6.600	1.1323E+06	0.	629.6	2.1271E+05	0.	0.	0.
1000	JUNC	6.600	1.1323E+06	0.	629.6	2.1271E+05	0.	0.	0.
1	JUNC	6.600	1.1323E+06	0.	629.6	2.1271E+05	0.	0.	0.
2	JUNC	6.600	1.1323E+06	0.	629.6	2.1271E+05	0.	0.	0.
3	JUNC	6.600	1.1323E+06	0.	629.6	2.1271E+05	0.	0.	0.
4	JUNC	6.600	1.1323E+06	0.	629.6	2.1271E+05	0.	0.	0.
5	JUNC	6.600	1.1323E+06	0.	629.6	2.1271E+05	0.	0.	0.
6	JUNC	6.600	1.1323E+06	0.	629.6	2.1271E+05	0.	0.	0.
7	JUNC	6.600	1.1323E+06	0.	629.6	2.1271E+05	0.	0.	0.
8	JUNC	6.600	1.1323E+06	0.	629.6	2.1271E+05	0.	0.	0.
9	JUNC	6.600	1.1323E+06	0.	629.6	2.1271E+05	0.	0.	0.
100	JUNC	6.600	1.1323E+06	0.	629.6	2.1271E+05	0.	0.	0.
99	PLEN	29.00	1.1323E+06	0.1000	77.55	4.3165E+05	0.	0.	0.

FLUID SUBMODEL NAME = CPL ; FLUID NO. = 717

MAX TIME STEP = 0. ; LIMITING MESSAGE = (NO SOLUTION STEP TAKEN YET)
LAST TIME STEP = 0. VS. DTMAXF/DTMINF = 10.0000 / 1.000000E-03; AVERAGE TIME STEP = 0.
PROBLEM TIME TIMEN = 0. VS. TIMEND = 120.000

TIE PARAMETER TABULATION FOR SUBMODEL CPL

TIE	TYPE	UA	QTIE	LUMP	TEMP.	NODE	TEMP.	PATH 1	FRACT	PATH 2	FRACT
35	BTN	0.	0.	35	6.600	EVAP.1500	6.600	35	1.0000		
40	BTN	0.	0.	40	6.600	EVAP.2000	6.600	40	4.0000		
1000	BTM	0.	0.	1000	6.600	EVAP.1000	6.600				
1	BTNC	0.	0.	1	6.600	PLATE.1	6.600	1	1.0000	2	0.5000
2	BTNC	0.	0.	2	6.600	PLATE.2	6.600	2	0.5000	3	0.5000
3	BTNC	0.	0.	3	6.600	PLATE.3	6.600	3	0.5000	4	0.5000
4	BTNC	0.	0.	4	6.600	PLATE.4	6.600	4	0.5000	5	0.5000
5	BTNC	0.	0.	5	6.600	PLATE.5	6.600	5	0.5000	6	0.5000
6	BTNC	0.	0.	6	6.600	PLATE.6	6.600	6	0.5000	7	0.5000
7	BTNC	0.	0.	7	6.600	PLATE.7	6.600	7	0.5000	8	0.5000
8	BTNC	0.	0.	8	6.600	PLATE.8	6.600	8	0.5000	9	0.5000
9	BTNC	0.	0.	9	6.600	PLATE.9	6.600	9	0.5000	10	1.0000

MODEL = SINDA85 FLUINT SAMPLE MODEL 5 - CPL GET AWAY SPECIAL (GAS) EXPERIMENT
FWDBCK

FLUID SUBMODEL NAME = CPL ; FLUID NO. = 717

MAX TIME STEP = 0. ; LIMITING MESSAGE = (NO SOLUTION STEP TAKEN YET)
LAST TIME STEP = 0. VS. DTMAXF/DTMINF = 10.0000 / 1.000000E-03; AVERAGE TIME STEP = 0.
PROBLEM TIME TIMEN = 0. VS. TIMEND = 120.000

PATH PARAMETER TABULATION FOR SUBMODEL CPL

PATH	TYPE	LMP 1	LMP 2	DUP I	DUP J	STAT	XL UPSTRM	FLOWRATE	DELTA PRES	FLOW REGIMES OR REYNOLDS NO.	
30	STUBE	30	35	45.	45.	NORM	0.	0.	0.	0.	0.
35	STUBE	35	40	1.0	2.0	NORM	0.	0.	0.	0.	0.
40	STUBE	40	50	1.0	1.0	NORM	0.	0.	0.	0.	0.
300	CAPIL	10	20	2.0	1.0	DLS	0.	0.	0.		
1000	NULL	20	1000	1.0	1.0	DLS	0.	0.	0.		
2000	NULL	1000	30	1.0	1.0	DLS	0.	0.	0.		
1	STUBE	50	1	1.0	1.0	NORM	0.	0.	0.	0.	0.
2	STUBE	1	2	1.0	1.0	NORM	0.	0.	0.	0.	0.
3	STUBE	2	3	1.0	1.0	NORM	0.	0.	0.	0.	0.
4	STUBE	3	4	1.0	1.0	NORM	0.	0.	0.	0.	0.
5	STUBE	4	5	1.0	1.0	NORM	0.	0.	0.	0.	0.
6	STUBE	5	6	1.0	1.0	NORM	0.	0.	0.	0.	0.
7	STUBE	6	7	1.0	1.0	NORM	0.	0.	0.	0.	0.
8	STUBE	7	8	1.0	1.0	NORM	0.	0.	0.	0.	0.
9	STUBE	8	9	1.0	1.0	NORM	0.	0.	0.	0.	0.
10	STUBE	9	100	1.0	1.0	NORM	0.	0.	0.	0.	0.
100	STUBE	100	10	1.0	1.0	NORM	0.	0.	0.	0.	0.
99	STUBE	10	99	1.0	1.0	RLS	0.	0.	0.	0.	0.

MODEL = SINDA85 FLUINT SAMPLE MODEL 5 - CPL GET AWAY SPECIAL (GAS) EXPERIMENT
FWDBCK

SUBMODEL NAME = PLATE

	CALCULATED			ALLOWED
MAX DIFF DELTA T PER ITER	DRLXCC(	0)=	0.	VS. DRLXCA= 1.000000E-02
MAX ARITH DELTA T PER ITER	ARLXCC(	0)=	0.	VS. ARLXCA= 1.000000E-02
MAX DIFF DEL T PER TIME STEP	DTMPCC(	0)=	0.	VS. DTMPCA= 1.000000E+30
MAX ARITH DEL T PER TIME STEP	ATMPCC(	0)=	0.	VS. ATMPCA= 1.000000E+30
MIN STABILITY CRITERIA	CSGMIN(	0)=	0.	
MAX STABILITY CRITERIA	CSGMAX(	0)=	0.	
NUMBER OF ITERATIONS	LOOPCT	=	0	VS. NLOOPT= 100
PROBLEM TIME	TIMEN	=	0.	VS. TIMEND= 120.000
MEAN PROBLEM TIME	TIMEM	=	0.	
AVERAGE TIME STEP USED SINCE LAST OUTPUT		=	0.	VS. DTIMEI= 0.

DIFFUSION NODES IN INPUT NODE NUMBER ORDER

T	1=	6.6000	T	2=	6.6000	T	3=	6.6000	T	4=	6.6000	T	5=	6.6000	T	6=	6.6000
T	7=	6.6000	T	8=	6.6000	T	9=	6.6000	T	11=	6.6000	T	12=	6.6000	T	13=	6.6000
T	14=	6.6000	T	15=	6.6000	T	16=	6.6000	T	17=	6.6000	T	18=	6.6000	T	19=	6.6000

ARITHMETIC NODES IN INPUT NODE NUMBER ORDER
↔NONE↔

HEATER NODES IN INPUT NODE NUMBER ORDER
↔NONE↔

BOUNDARY NODES IN INPUT NODE NUMBER ORDER

T 300= 6.6000

MODEL = SINDA85
FWDBCK

FLUINT SAMPLE MODEL 5 - CPL GET AWAY SPECIAL (GAS) EXPERIMENT

SUBMODEL NAME = EVAP

	CALCULATED			ALLOWED	
MAX DIFF DELTA T PER ITER	DRLXCC(	0)=	0.	VS. DRLXCA=	1.000000E-02
MAX ARITH DELTA T PER ITER	ARLXCC(	0)=	0.	VS. ARLXCA=	1.000000E-02
MAX DIFF DEL T PER TIME STEP	DTMPCC(	0)=	0.	VS. DTMPCA=	23.4009
MAX ARITH DEL T PER TIME STEP	ATMPCC(	0)=	0.	VS. ATMPCA=	1.000000E+30
MIN STABILITY CRITERIA	CSGMIN(	0)=	0.		
MAX STABILITY CRITERIA	CSGMAX(	0)=	0.		
NUMBER OF ITERATIONS	LOOPCT	=	0	VS. NLOOPT=	100
PROBLEM TIME	TIMEN	=	0.	VS. TIMEND=	120.000
MEAN PROBLEM TIME	TIMEM	=	0.		
AVERAGE TIME STEP USED SINCE LAST OUTPUT		=	0.	VS. DTIMEI=	0.

DIFFUSION NODES IN INPUT NODE NUMBER ORDER

T 1000= 6.6000 T 2000= 6.6000

ARITHMETIC NODES IN INPUT NODE NUMBER ORDER

T 1500= 6.6000

HEATER NODES IN INPUT NODE NUMBER ORDER
↔NONE↔

BOUNDARY NODES IN INPUT NODE NUMBER ORDER
↔NONE↔

SUBMODEL NAME = EVAP

ARLXCA =	1.000000E-02	ARLXCC =	0.	ATMPCA =	1.000000E+30	ATMPCC =	0.	BACKUP =	0.		
CSGFAC =	0.	CSGMAX =	0.	CSGMIN =	0.	DRLXCA =	1.000000E-02	DRLXCC =	0.		
DTIMEB =	1.000000E+30	DTIMEI =	0.	DTIMEL =	0.	DTIMEU =	0.	DTMPCA =	23.4009		
DTMPCC =	0.	EBALNA =	0.	EBALNC =	0.	EBALSA =	1.000000E-02	EBALSC =	0.		
ESUMIS =	0.	ESUMOS =	0.	EXTLIM =	50.0000	ITHOLD =	10	NARLXN =	0		
NATMPN =	0	NCGMAN =	0	NCSGMN =	0	NDRLXN =	0	NDTMPN =	0		
NEBALN =	0	NLOOPT =	100	ITEROT =	0	ITERXT =	3	OPEITR =	0		
OUTPUT =	6.00000										

GLOBAL CONTROL CONSTANTS

ABSZRO =	-273.160	SIGMA =	1.00000	TIMEM =	0.	TIMEN =	0.	TIMEND =	120.000
TIMEO =	0.								

*** WARNING *** CAPPMP HAS DEPRIMED
MODEL: CPL , (JUNCTION 1000) LOOPCT = 0, TIMEN = 0.
EXCESS LIQUID IN HIGHER PRESSURE LUMP
QUALITY OF LUMP 30 IS 0. VS. 0.9500 , PRESSURE DIFFERENCE = 0.

*** WARNING *** CAPIL HAS DEPRIMED
MODEL: CPL , (CONNECTOR 300) LOOPCT = 0, TIMEN = 0.
EXCESS LIQUID IN HIGHER PRESSURE LUMP
QUALITY OF LUMP 20 IS 0. VS. 0.9900 , PRESSURE DIFFERENCE = 0.

MODEL = SINDA85 FLUINT SAMPLE MODEL 5 - CPL GET AWAY SPECIAL (GAS) EXPERIMENT
FWDBCK

FLUID SUBMODEL NAME = CPL ; FLUID NO. = 717

MAX TIME STEP = 6.639614E-02 ; LIMITING TANK = 40 REASON = TEMP./DENSITY CHANGE LIMIT
LAST TIME STEP = 1.526785E-02 VS. DTMAXF/DTMINF = 10.0000 / 1.000000E-03; AVERAGE TIME STEP = 6.941558E-02
PROBLEM TIME TIMEN = 18.0000 VS. TIMEND = 120.000

LUMP PARAMETER TABULATION FOR SUBMODEL CPL

LUMP	TYPE	TEMP	PRESSURE	QUALITY	DENSITY	ENTHALPY	HEAT RATE	MASS RATE	ENERGY RATE
10	TANK	6.607	1.1323E+06	0.	629.6	2.1274E+05	0.	-4.8058E-08	0.2466
20	TANK	6.600	1.1321E+06	0.	629.6	2.1271E+05	0.	-1.4108E-08	1.8120E-04
40	TANK	19.26	1.1323E+06	0.	611.5	2.7119E+05	-24.32	-1.1098E-04	186.2
30	JUNC	29.01	1.1323E+06	1.000	8.782	1.4660E+06	0.	0.	0.
35	JUNC	29.00	1.1323E+06	0.9965	8.813	1.4619E+06	-0.4105	7.2760E-12	2.0981E-05
50	JUNC	19.26	1.1323E+06	0.	611.5	2.7119E+05	0.	0.	0.
1000	JUNC	29.00	1.1322E+06	1.000	8.782	1.4660E+06	126.6	0.	-5.4169E-04
1	JUNC	9.560	1.1323E+06	0.	625.4	2.2633E+05	-14.04	0.	0.
2	JUNC	7.387	1.1323E+06	0.	628.5	2.1633E+05	-3.130	0.	0.
3	JUNC	6.916	1.1323E+06	0.	629.1	2.1417E+05	-0.6765	0.	0.
4	JUNC	6.816	1.1323E+06	0.	629.3	2.1370E+05	-0.1443	0.	0.
5	JUNC	6.795	1.1323E+06	0.	629.3	2.1361E+05	-3.0472E-02	0.	0.
6	JUNC	6.787	1.1323E+06	0.	629.3	2.1357E+05	-1.0422E-02	0.	0.
7	JUNC	6.786	1.1323E+06	0.	629.3	2.1357E+05	-2.2278E-03	0.	0.
8	JUNC	6.785	1.1323E+06	0.	629.3	2.1356E+05	-7.2479E-04	0.	0.
9	JUNC	6.785	1.1323E+06	0.	629.3	2.1356E+05	2.3651E-04	2.9104E-11	2.6703E-04
100	JUNC	6.785	1.1323E+06	0.	629.3	2.1356E+05	0.	-2.9104E-11	-7.6294E-06
99	PLEN	29.00	1.1323E+06	0.1000	77.55	4.3165E+05	0.	1.1106E-04	23.63

MODEL = SINDA85 FLUINT SAMPLE MODEL 5 - CPL GET AWAY SPECIAL (GAS) EXPERIMENT
FWDBCK

FLUID SUBMODEL NAME = CPL ; FLUID NO. = 717

MAX TIME STEP = 6.639614E-02 ; LIMITING TANK = 40 REASON = TEMP./DENSITY CHANGE LIMIT
LAST TIME STEP = 1.526785E-02 VS. DTMAXF/DTMINF = 10.0000 / 1.000000E-03; AVERAGE TIME STEP = 6.941558E-02
PROBLEM TIME TIMEN = 18.0000 VS. TIMEND = 120.000

TIE PARAMETER TABULATION FOR SUBMODEL CPL

TIE	TYPE	UA	QTIE	LUMP	TEMP.	NODE	TEMP.	PATH 1	FRACT	PATH 2	FRACT
35	ETN	43.76	-0.4105	35	29.00	EVAP.1500	28.99	35	1.0000		
40	ETN	2.890	-24.32	40	19.26	EVAP.2000	10.81	40	4.0000		
1000	ETM	15.96	126.6	1000	29.00	EVAP.1000	36.95				
1	ETMC	6.737	-14.04	1	9.560	PLATE.1	7.347	1	1.0000	2	0.5000
2	ETMC	6.803	-3.130	2	7.387	PLATE.2	6.777	2	0.5000	3	0.5000
3	ETMC	6.817	-0.6765	3	6.916	PLATE.3	6.659	3	0.5000	4	0.5000
4	ETMC	6.820	-0.1443	4	6.816	PLATE.4	6.636	4	0.5000	5	0.5000
5	ETMC	6.821	-3.0472E-02	5	6.795	PLATE.5	6.631	5	0.5000	6	0.5000
6	ETMC	6.821	-1.0422E-02	6	6.787	PLATE.6	6.627	6	0.5000	7	0.5000
7	ETMC	6.821	-2.2278E-03	7	6.786	PLATE.7	6.626	7	0.5000	8	0.5000
8	ETMC	6.821	-7.2479E-04	8	6.785	PLATE.8	6.626	8	0.5000	9	0.5000
9	ETMC	6.821	2.3651E-04	9	6.785	PLATE.9	6.626	9	0.5000	10	1.0000

MODEL = SINDA85 FLUINT SAMPLE MODEL 5 - CPL GET AWAY SPECIAL (GAS) EXPERIMENT
FWDBCK

FLUID SUBMODEL NAME = CPL ; FLUID NO. = 717

MAX TIME STEP = 6.639614E-02 ; LIMITING TANK = 40 REASON = TEMP./DENSITY CHANGE LIMIT
LAST TIME STEP = 1.526785E-02 VS. DTMAXF/DTMINF = 10.0000 / 1.000000E-03; AVERAGE TIME STEP = 6.941558E-02
PROBLEM TIME TIMEN = 18.0000 VS. TIMEND = 120.000

PATH PARAMETER TABULATION FOR SUBMODEL CPL

PATH	TYPE	LMP 1	LMP 2	DUP I	DUP J	STAT	XL UPSTRM	FLOWRATE	DELTA PRES	FLOW REGIMES OR REYNOLDS NO.	
30	STUBE	30	35	45.	45.	NORM	1.000	2.2448E-06	1.530	202.80	
35	STUBE	35	40	1.0	2.0	NORM	0.9965	1.0102E-04	0.2015	UNKNOWN	
40	STUBE	40	50	1.0	1.0	NORM	0.	3.1301E-04	1.6759E-02	184.72	
300	CAPIL	10	20	2.0	1.0	DLS	0.	1.0100E-04	177.4		
1000	NULL	20	1000	1.0	1.0	LS	0.	1.0102E-04	-106.2		
2000	NULL	1000	30	1.0	1.0	RLS	1.000	1.0102E-04	-106.2		
1	STUBE	50	1	1.0	1.0	NORM	0.	3.1301E-04	1.611	173.72	155.94
2	STUBE	1	2	1.0	1.0	NORM	0.	3.1301E-04	3.317	155.94	152.21
3	STUBE	2	3	1.0	1.0	NORM	0.	3.1301E-04	3.399	152.21	151.41
4	STUBE	3	4	1.0	1.0	NORM	0.	3.1301E-04	3.366	151.41	151.24
5	STUBE	4	5	1.0	1.0	NORM	0.	3.1301E-04	3.410	151.24	151.21
6	STUBE	5	6	1.0	1.0	NORM	0.	3.1301E-04	3.368	151.21	151.20
7	STUBE	6	7	1.0	1.0	NORM	0.	3.1301E-04	3.410	151.20	151.19
8	STUBE	7	8	1.0	1.0	NORM	0.	3.1301E-04	3.368	151.19	151.19
9	STUBE	8	9	1.0	1.0	NORM	0.	3.1301E-04	3.410	151.19	151.19
10	STUBE	9	100	1.0	1.0	NORM	0.	3.1301E-04	1.684	151.19	151.19
100	STUBE	100	10	1.0	1.0	NORM	0.	3.1301E-04	2.897	151.19	150.89
99	STUBE	10	99	1.0	1.0	RLS	0.	1.1106E-04	4.385	53.536	68.634

MODEL = SINDA85 FLUINT SAMPLE MODEL 5 - CPL GET AWAY SPECIAL (GAS) EXPERIMENT
FWDBCK

SUBMODEL NAME = PLATE

	CALCULATED			ALLOWED	
MAX DIFF DELTA T PER ITER	DRLXCC(PLATE	1)=	3.051758E-05 VS.	DRLXCA=	1.000000E-02
MAX ARITH DELTA T PER ITER	ARLXCC(	0)=	0. VS.	ARLXCA=	1.000000E-02
MAX DIFF DEL T PER TIME STEP	DTMPCC(PLATE	1)=	5.218506E-03 VS.	DTMPCA=	1.000000E+30
MAX ARITH DEL T PER TIME STEP	ATMPCC(	0)=	0. VS.	ATMPCA=	1.000000E+30
MIN STABILITY CRITERIA	CSGMIN(PLATE	1)=	0.397076		
MAX STABILITY CRITERIA	CSGMAX(PLATE	19)=	7.60046		
NUMBER OF ITERATIONS	LOOPCT	=	2 VS.	NLOOPT=	100
PROBLEM TIME	TIMEN	=	18.0000 VS.	TIMEND=	120.000
MEAN PROBLEM TIME	TIMEM	=	17.9924		
AVERAGE TIME STEP USED SINCE LAST OUTPUT		=	9.836066E-02 VS.	DTIMEI=	0.

DIFFUSION NODES IN INPUT NODE NUMBER ORDER

T	1= 7.3465	T	2= 6.7769	T	3= 6.6591	T	4= 6.6357	T	5= 6.6310	T	6= 6.6266			
T	7= 6.6263	T	8= 6.6261	T	9= 6.6261	T	11= 6.6987	T	12= 6.6455	T	13= 6.6335			
T	14= 6.6318	T	15= 6.6317	T	16= 6.6275	T	17= 6.6274	T	18= 6.6274	T	19= 6.6274			

ARITHMETIC NODES IN INPUT NODE NUMBER ORDER
↔NONE↔

HEATER NODES IN INPUT NODE NUMBER ORDER
↔NONE↔

BOUNDARY NODES IN INPUT NODE NUMBER ORDER

T 300= 6.6743

MODEL = SINDA85 FLUINT SAMPLE MODEL 5 - CPL GET AWAY SPECIAL (GAS) EXPERIMENT
FWDBCK

SUBMODEL NAME = EVAP

	CALCULATED			ALLOWED	
MAX DIFF DELTA T PER ITER	DRLXCC(EVAP	2000)=	0.	VS. DRLXCA=	1.000000E-02
MAX ARITH DELTA T PER ITER	ARLXCC(EVAP	1500)=	0.	VS. ARLXCA=	1.000000E-02
MAX DIFF DEL T PER TIME STEP	DTMPCC(EVAP	2000)=	2.587891E-02	VS. DTMPCA=	5.00000
MAX ARITH DEL T PER TIME STEP	ATMPCC(EVAP	1500)=	2.441406E-04	VS. ATMPCA=	1.000000E+30
MIN STABILITY CRITERIA	CSGMIN(EVAP	2000)=	4.88561		
MAX STABILITY CRITERIA	CSGMAX(EVAP	1000)=	8.60554		
NUMBER OF ITERATIONS	LOOPCT	=	2	VS. NLOOPT=	100
PROBLEM TIME	TIMEN	=	18.0000	VS. TIMEND=	120.000
MEAN PROBLEM TIME	TIMEM	=	17.9924		
AVERAGE TIME STEP USED SINCE LAST OUTPUT		=	9.836066E-02	VS. DTIMEI=	0.

DIFFUSION NODES IN INPUT NODE NUMBER ORDER

T 1000= 36.948 T 2000= 10.813

ARITHMETIC NODES IN INPUT NODE NUMBER ORDER

T 1500= 28.993

HEATER NODES IN INPUT NODE NUMBER ORDER
--NONE--

BOUNDARY NODES IN INPUT NODE NUMBER ORDER
--NONE--

MODEL = SINDA85 FLUINT SAMPLE MODEL 5 - CPL GET AWAY SPECIAL (GAS) EXPERIMENT
FWDBCK

FLUID SUBMODEL NAME = CPL ; FLUID NO. = 717

MAX TIME STEP = 7.958720E-02 ; LIMITING TANK = 40 REASON = FR CHANGE LIMIT IN PATH 40
LAST TIME STEP = 6.847715E-02 VS. DTMAXF/DTMINF = 10.0000 / 1.000000E-05; AVERAGE TIME STEP = 5.282040E-02
PROBLEM TIME TIMEN = 24.0000 VS. TIMEND = 120.000

LUMP PARAMETER TABULATION FOR SUBMODEL CPL

LUMP	TYPE	TEMP	PRESSURE	QUALITY	DENSITY	ENTHALPY	HEAT RATE	MASS RATE	ENERGY RATE
10	TANK	7.363	1.1323E+06	0.	628.5	2.1622E+05	0.	-7.3182E-06	-2.817
20	TANK	6.614	1.1320E+06	0.	629.6	2.1278E+05	0.	-2.5397E-07	0.4686
40	TANK	29.00	1.1324E+06	0.5938	14.64	9.9921E+05	-3.049	-1.9819E-04	-58.74
30	JUNC	29.01	1.1324E+06	1.000	8.782	1.4660E+06	0.	0.	0.
35	JUNC	29.41	1.1324E+06	1.000	8.763	1.4672E+06	0.1857	0.	1.9073E-06
50	JUNC	29.00	1.1324E+06	0.5938	14.64	9.9921E+05	0.	0.	0.
1000	JUNC	29.00	1.1322E+06	1.000	8.781	1.4660E+06	190.6	0.	6.0654E-02
1	JUNC	29.00	1.1324E+06	0.2006	41.36	5.4725E+05	-227.0	5.8208E-11	1.5259E-05
2	JUNC	29.00	1.1324E+06	5.6662E-03	432.7	3.2324E+05	-112.5	0.	0.
3	JUNC	15.40	1.1323E+06	0.	617.1	2.5330E+05	-35.13	0.	0.
4	JUNC	10.09	1.1323E+06	0.	624.7	2.2878E+05	-12.32	0.	0.
5	JUNC	8.122	1.1323E+06	0.	627.5	2.1971E+05	-4.557	0.	0.
6	JUNC	7.241	1.1323E+06	0.	628.7	2.1566E+05	-2.055	0.	-1.9051E-02
7	JUNC	7.041	1.1323E+06	0.	629.0	2.1474E+05	-0.9103	0.	-0.4485
8	JUNC	6.907	1.1323E+06	0.	629.2	2.1412E+05	-0.4870	-5.8208E-11	-0.1775
9	JUNC	6.828	1.1323E+06	0.	629.3	2.1376E+05	-0.2742	0.	-9.2995E-02
100	JUNC	6.828	1.1323E+06	0.	629.3	2.1376E+05	0.	-5.8208E-11	-1.5259E-05
99	PLEN	29.00	1.1323E+06	0.1000	77.55	4.3165E+05	0.	2.0602E-04	44.55

MODEL = SINDA85 FLUINT SAMPLE MODEL 5 - CPL GET AWAY SPECIAL (GAS) EXPERIMENT
FWDBCK

FLUID SUBMODEL NAME = CPL ; FLUID NO. = 717

MAX TIME STEP = 7.958720E-02 ; LIMITING TANK = 40 REASON = FR CHANGE LIMIT IN PATH 40
LAST TIME STEP = 6.847715E-02 VS. DTMAXF/DTMINF = 10.0000 / 1.000000E-05; AVERAGE TIME STEP = 5.282040E-02
PROBLEM TIME TIMEN = 24.0000 VS. TIMEND = 120.000

TIE PARAMETER TABULATION FOR SUBMODEL CPL

TIE	TYPE	UA	QTIE	LUMP	TEMP.	NODE	TEMP.	PATH 1	FRACT	PATH 2	FRACT
35	BTN	5.8327E-02	0.1857	35	29.41	EVAP.1500	32.61	35	1.0000		
40	BTN	23.90	-3.049	40	29.00	EVAP.2000	28.89	40	4.0000		
1000	BTM	15.96	190.6	1000	29.00	EVAP.1000	40.98				
1	BTNC	34.27	-227.0	1	29.00	PLATE.1	22.41	1	1.0000	2	0.5000
2	BTNC	7.323	-112.5	2	29.00	PLATE.2	13.78	2	0.5000	3	0.5000
3	BTNC	6.556	-35.13	3	15.40	PLATE.3	10.00	3	0.5000	4	0.5000
4	BTNC	6.710	-12.32	4	10.09	PLATE.4	8.424	4	0.5000	5	0.5000
5	BTNC	6.767	-4.557	5	8.122	PLATE.5	7.711	5	0.5000	6	0.5000
6	BTNC	6.793	-2.055	6	7.241	PLATE.6	7.241	6	0.5000	7	0.5000
7	BTNC	6.804	-0.9103	7	7.041	PLATE.7	7.041	7	0.5000	8	0.5000
8	BTNC	6.810	-0.4870	8	6.907	PLATE.8	6.907	8	0.5000	9	0.5000
9	BTNC	6.814	-0.2742	9	6.828	PLATE.9	6.829	9	0.5000	10	1.0000

MODEL = SINDA85 FLUINT SAMPLE MODEL 5 - CPL GET AWAY SPECIAL (GAS) EXPERIMENT
FWDBCK

FLUID SUBMODEL NAME = CPL ; FLUID NO. = 717

MAX TIME STEP = 7.958720E-02 ; LIMITING TANK = 40 REASON = FR CHANGE LIMIT IN PATH 40
LAST TIME STEP = 6.847715E-02 VS. DTMAXF/DTMINF = 10.0000 / 1.000000E-05; AVERAGE TIME STEP = 5.282040E-02
PROBLEM TIME TIMEN = 24.0000 VS. TIMEND = 120.000

PATH PARAMETER TABULATION FOR SUBMODEL CPL

PATH	TYPE	LMP 1	LMP 2	DUP I	DUP J	STAT	XL UPSTRM	FLOWRATE	DELTA PRES	FLOW REGIMES OR REYNOLDS NO.	
30	STUBE	30	35	45.	45.	NORM	1.000	3.3793E-06	2.499	305.29	
35	STUBE	35	40	1.0	2.0	NORM	1.000	1.5207E-04	0.5605	2835.6	
40	STUBE	40	50	1.0	1.0	NORM	0.5938	5.0233E-04	0.8220	UNKNOWN	
300	CAPIL	10	20	2.0	1.0	DLS	0.	1.5181E-04	264.8		
1000	NULL	20	1000	1.0	1.0	LS	0.	1.5207E-04	-157.2		
2000	NULL	1000	30	1.0	1.0	RLS	1.000	1.5207E-04	-157.2		
1	STUBE	50	1	1.0	1.0	NORM	0.5938	5.0233E-04	-3.070	UNKNOWN	UNKNOWN
2	STUBE	1	2	1.0	1.0	NORM	0.2006	5.0233E-04	4.221	UNKNOWN	UNKNOWN
3	STUBE	2	3	1.0	1.0	NORM	5.6662E-03	5.0233E-04	5.056	UNKNOWN	267.10
4	STUBE	3	4	1.0	1.0	NORM	0.	5.0233E-04	5.108	267.10	251.76
5	STUBE	4	5	1.0	1.0	NORM	0.	5.0233E-04	5.381	251.76	246.28
6	STUBE	5	6	1.0	1.0	NORM	0.	5.0233E-04	5.338	246.28	243.87
7	STUBE	6	7	1.0	1.0	NORM	0.	5.0233E-04	5.477	243.87	243.33
8	STUBE	7	8	1.0	1.0	NORM	0.	5.0233E-04	5.383	243.33	242.97
9	STUBE	8	9	1.0	1.0	NORM	0.	5.0233E-04	5.498	242.97	242.75
10	STUBE	9	100	1.0	1.0	NORM	0.	5.0233E-04	2.697	242.75	242.75
100	STUBE	100	10	1.0	1.0	NORM	0.	5.0233E-04	4.628	242.75	244.21
99	STUBE	10	99	1.0	1.0	RLS	0.	2.0602E-04	8.125	100.16	127.32

MODEL = SINDA85 FLUINT SAMPLE MODEL 5 - CPL GET AWAY SPECIAL (GAS) EXPERIMENT
FWDBCK

SUBMODEL NAME = PLATE

	CALCULATED			ALLOWED	
MAX DIFF DELTA T PER ITER	DRLXCC(PLATE	1)=	2.990723E-03 VS.	DRLXCA=	1.000000E-02
MAX ARITH DELTA T PER ITER	ARLXCC(	0)=	0. VS.	ARLXCA=	1.000000E-02
MAX DIFF DEL T PER TIME STEP	DTMPCC(PLATE	2)=	0.118164 VS.	DTMPCA=	1.000000E+30
MAX ARITH DEL T PER TIME STEP	ATMPCC(	0)=	0. VS.	ATMPCA=	1.000000E+30
MIN STABILITY CRITERIA	CSGMIN(PLATE	1)=	0.185088		
MAX STABILITY CRITERIA	CSGMAX(PLATE	19)=	7.60046		
NUMBER OF ITERATIONS	LOOPCT	=	2 VS.	NLOOPT=	100
PROBLEM TIME	TIMEN	=	24.0000 VS.	TIMEND=	120.000
MEAN PROBLEM TIME	TIMEM	=	23.9658		
AVERAGE TIME STEP USED SINCE LAST OUTPUT		=	3.076923E-02 VS.	DTIMEI=	0.

DIFFUSION NODES IN INPUT NODE NUMBER ORDER

T	1=	22.405	T	2=	13.778	T	3=	10.004	T	4=	8.4241	T	5=	7.7113	T	6=	7.2410
T	7=	7.0410	T	8=	6.9069	T	9=	6.8286	T	11=	9.4080	T	12=	8.1463	T	13=	7.6132
T	14=	7.2989	T	15=	7.1402	T	16=	6.8984	T	17=	6.8363	T	18=	6.7710	T	19=	6.7377

ARITHMETIC NODES IN INPUT NODE NUMBER ORDER
↔NONE↔

HEATER NODES IN INPUT NODE NUMBER ORDER
↔NONE↔

BOUNDARY NODES IN INPUT NODE NUMBER ORDER

T 300= 6.6989

MODEL = SINDA85 FLUINT SAMPLE MODEL 5 - CPL GET AWAY SPECIAL (GAS) EXPERIMENT
FWDBCK

SUBMODEL NAME = EVAP

	CALCULATED			ALLOWED	
MAX DIFF DELTA T PER ITER	DRLXCC(EVAP	2000)=	0. VS.	DRLXCA=	1.000000E-02
MAX ARITH DELTA T PER ITER	ARLXCC(EVAP	1500)=	0. VS.	ARLXCA=	1.000000E-02
MAX DIFF DEL T PER TIME STEP	DTMPCC(EVAP	1000)=	3.186035E-02 VS.	DTMPCA=	5.00000
MAX ARITH DEL T PER TIME STEP	ATMPCC(EVAP	1500)=	8.911133E-03 VS.	ATMPCA=	1.000000E+30
MIN STABILITY CRITERIA	CSGMIN(EVAP	2000)=	0.622964		
MAX STABILITY CRITERIA	CSGMAX(EVAP	1000)=	8.60554		
NUMBER OF ITERATIONS	LOOPCT	=	2 VS.	NLOOPT=	100
PROBLEM TIME	TIMEN	=	24.0000 VS.	TIMEND=	120.000
MEAN PROBLEM TIME	TIMEM	=	23.9658		
AVERAGE TIME STEP USED SINCE LAST OUTPUT		=	3.076923E-02 VS.	DTIMEI=	0.

DIFFUSION NODES IN INPUT NODE NUMBER ORDER

T 1000= 40.978 T 2000= 28.887

ARITHMETIC NODES IN INPUT NODE NUMBER ORDER

T 1500= 32.611

HEATER NODES IN INPUT NODE NUMBER ORDER
↔NONE↔

BOUNDARY NODES IN INPUT NODE NUMBER ORDER
↔NONE↔

MODEL = SINDA85
FWDBCK

FLUINT SAMPLE MODEL 5 - CPL GET AWAY SPECIAL (GAS) EXPERIMENT

FLUID SUBMODEL NAME = CPL ; FLUID NO. = 717

MAX TIME STEP = 1.25169 ; LIMITING TANK = 40 REASON = FR CHANGE LIMIT IN PATH 40
LAST TIME STEP = 0.276850 VS. DTMAXF/DTMINF = 10.0000 / 1.000000E-03; AVERAGE TIME STEP = 0.232895
PROBLEM TIME TIMEN = 30.0000 VS. TIMEND = 120.000

LUMP PARAMETER TABULATION FOR SUBMODEL CPL

LUMP	TYPE	TEMP	PRESSURE	QUALITY	DENSITY	ENTHALPY	HEAT RATE	MASS RATE	ENERGY RATE
10	TANK	7.469	1.1323E+06	0.	628.4	2.1671E+05	0.	5.5047E-07	1.540
20	TANK	6.666	1.1320E+06	0.	629.5	2.1301E+05	0.	-3.1544E-07	0.5867
40	TANK	29.32	1.1323E+06	1.000	8.767	1.4670E+06	-0.1326	-6.2108E-08	-6.3075E-02
30	JUNC	29.01	1.1323E+06	1.000	8.782	1.4660E+06	0.	0.	0.
35	JUNC	29.47	1.1323E+06	1.000	8.760	1.4674E+06	0.2513	0.	1.5259E-05
50	JUNC	29.32	1.1323E+06	1.000	8.767	1.4670E+06	0.	0.	0.
1000	JUNC	29.00	1.1322E+06	1.000	8.781	1.4660E+06	222.3	0.	4.2725E-04
1	JUNC	29.00	1.1323E+06	0.5110	16.95	9.0401E+05	-199.8	2.9104E-11	4.5776E-05
2	JUNC	29.00	1.1323E+06	2.8515E-02	205.2	3.4950E+05	-196.8	0.	0.
3	JUNC	16.22	1.1323E+06	0.	615.9	2.5709E+05	-32.80	0.	0.
4	JUNC	10.70	1.1323E+06	0.	623.8	2.3157E+05	-9.054	0.	0.
5	JUNC	9.294	1.1323E+06	0.	625.8	2.2510E+05	-2.296	2.9104E-11	7.6294E-06
6	JUNC	8.670	1.1323E+06	0.	626.7	2.2223E+05	-1.019	0.	0.
7	JUNC	8.488	1.1323E+06	0.	626.9	2.2139E+05	-0.2971	0.	0.
8	JUNC	8.383	1.1323E+06	0.	627.1	2.2091E+05	-0.1717	0.	0.
9	JUNC	8.340	1.1323E+06	0.	627.2	2.2071E+05	-7.1068E-02	0.	0.
100	JUNC	8.340	1.1323E+06	0.	627.2	2.2071E+05	0.	0.	0.
99	PLEN	29.00	1.1323E+06	0.1000	77.55	4.3165E+05	0.	1.4246E-07	3.0871E-02

MODEL = SINDA85
FWDBCK

FLUINT SAMPLE MODEL 5 - CPL GET AWAY SPECIAL (GAS) EXPERIMENT

FLUID SUBMODEL NAME = CPL ; FLUID NO. = 717

MAX TIME STEP = 1.25169 ; LIMITING TANK = 40 REASON = FR CHANGE LIMIT IN PATH 40
LAST TIME STEP = 0.276850 VS. DTMAXF/DTMINF = 10.0000 / 1.000000E-03; AVERAGE TIME STEP = 0.232895
PROBLEM TIME TIMEN = 30.0000 VS. TIMEND = 120.000

TIE PARAMETER TABULATION FOR SUBMODEL CPL

TIE	TYPE	UA	QTIE	LUMP	TEMP.	NODE	TEMP.	PATH 1	FRACT	PATH 2	FRACT
35	ETN	7.4984E-02	0.2513	35	29.47	EVAP.1500	32.85	35	1.0000		
40	ETN	0.4255	-0.1326	40	29.32	EVAP.2000	28.99	40	4.0000		
1000	ETM	15.96	222.3	1000	29.00	EVAP.1000	42.99				
1	ETNC	41.43	-199.8	1	29.00	PLATE.1	24.24	1	1.0000	2	0.5000
2	ETNC	26.57	-196.8	2	29.00	PLATE.2	21.89	2	0.5000	3	0.5000
3	ETNC	6.578	-32.80	3	16.22	PLATE.3	9.764	3	0.5000	4	0.5000
4	ETNC	6.746	-9.054	4	10.70	PLATE.4	7.806	4	0.5000	5	0.5000
5	ETNC	6.788	-2.296	5	9.294	PLATE.5	7.403	5	0.5000	6	0.5000
6	ETNC	6.807	-1.019	6	8.670	PLATE.6	6.945	6	0.5000	7	0.5000
7	ETNC	6.813	-0.2971	7	8.488	PLATE.7	6.865	7	0.5000	8	0.5000
8	ETNC	6.816	-0.1717	8	8.383	PLATE.8	6.775	8	0.5000	9	0.5000
9	ETNC	6.817	-7.1068E-02	9	8.340	PLATE.9	6.746	9	0.5000	10	1.0000

MODEL = SINDA85 FLUINT SAMPLE MODEL 5 - CPL GET AWAY SPECIAL (GAS) EXPERIMENT
FWDBCK

FLUID SUBMODEL NAME = CPL ; FLUID NO. = 717

MAX TIME STEP = 1.25169 ; LIMITING TANK = 40 REASON = FR CHANGE LIMIT IN PATH 40
LAST TIME STEP = 0.276850 VS. DTMAXF/DTMINF = 10.0000 / 1.000000E-03; AVERAGE TIME STEP = 0.232895
PROBLEM TIME TIMEN = 30.0000 VS. TIMEND = 120.000

PATH PARAMETER TABULATION FOR SUBMODEL CPL

PATH	TYPE	LMP 1	LMP 2	DUP I	DUP J	STAT	XL UPSTRM	FLOWRATE	DELTA PRES	FLOW REGIMES OR REYNOLDS NO.	
30	STUBE	30	35	45.	45.	NORM	1.000	3.9424E-06	3.021	356.16	
35	STUBE	35	40	1.0	2.0	NORM	1.000	1.7741E-04	0.4010	3307.3	
40	STUBE	40	50	1.0	1.0	NORM	1.000	3.5488E-04	2.018	3098.4	
300	CAPIL	10	20	2.0	1.0	DLS	0.	1.7709E-04	308.6		
1000	NULL	20	1000	1.0	1.0	LS	0.	1.7741E-04	-176.5		
2000	NULL	1000	30	1.0	1.0	RLS	1.000	1.7741E-04	-176.5		
1	STUBE	50	1	1.0	1.0	NORM	1.000	3.5488E-04	4.842	2914.0	UNKNOWN
2	STUBE	1	2	1.0	1.0	NORM	0.5110	3.5488E-04	1.783	UNKNOWN	UNKNOWN
3	STUBE	2	3	1.0	1.0	NORM	2.8515E-02	3.5488E-04	4.444	UNKNOWN	190.42
4	STUBE	3	4	1.0	1.0	NORM	0.	3.5488E-04	3.643	190.42	179.06
5	STUBE	4	5	1.0	1.0	NORM	0.	3.5488E-04	3.811	179.06	176.28
6	STUBE	5	6	1.0	1.0	NORM	0.	3.5488E-04	3.792	176.28	175.06
7	STUBE	6	7	1.0	1.0	NORM	0.	3.5488E-04	3.860	175.06	174.70
8	STUBE	7	8	1.0	1.0	NORM	0.	3.5488E-04	3.811	174.70	174.50
9	STUBE	8	9	1.0	1.0	NORM	0.	3.5488E-04	3.868	174.50	174.41
10	STUBE	9	100	1.0	1.0	NORM	0.	3.5488E-04	1.907	174.41	174.41
100	STUBE	100	10	1.0	1.0	NORM	0.	3.5488E-04	3.269	174.41	172.73
99	STUBE	10	99	1.0	1.0	RLS	0.	1.4246E-07	5.5832E-03	6.93370E-02	8.80401E-02

MODEL = SINDA85 FLUINT SAMPLE MODEL 5 - CPL GET AWAY SPECIAL (GAS) EXPERIMENT
FWDBCK

SUBMODEL NAME = PLATE

	CALCULATED			ALLOWED	
MAX DIFF DELTA T PER ITER	DRLXCC(PLATE	1)=	1.220703E-04	VS. DRLXCA=	1.000000E-02
MAX ARITH DELTA T PER ITER	ARLXCC(	0)=	0.	VS. ARLXCA=	1.000000E-02
MAX DIFF DEL T PER TIME STEP	DTMPCC(PLATE	2)=	0.296234	VS. DTMPCA=	1.000000E+30
MAX ARITH DEL T PER TIME STEP	ATMPCC(	0)=	0.	VS. ATMPCA=	1.000000E+30
MIN STABILITY CRITERIA	CSGMIN(PLATE	1)=	0.162505		
MAX STABILITY CRITERIA	CSGMAX(PLATE	19)=	7.60046		
NUMBER OF ITERATIONS	LOOPCT	=	3	VS. NLOOPT=	100
PROBLEM TIME	TIMEN	=	30.0000	VS. TIMEND=	120.000
MEAN PROBLEM TIME	TIMEM	=	29.8616		
AVERAGE TIME STEP USED SINCE LAST OUTPUT		=	0.181818	VS. DTIMEI=	0.

DIFFUSION NODES IN INPUT NODE NUMBER ORDER

T	1=	24.237	T	2=	21.892	T	3=	9.7637	T	4=	7.8062	T	5=	7.4026	T	6=	6.9448
T	7=	6.8652	T	8=	6.7747	T	9=	6.7464	T	11=	12.736	T	12=	10.967	T	13=	7.9431
T	14=	7.2989	T	15=	7.2972	T	16=	6.8875	T	17=	6.8490	T	18=	6.7642	T	19=	6.7427

ARITHMETIC NODES IN INPUT NODE NUMBER ORDER
↔NONE↔

HEATER NODES IN INPUT NODE NUMBER ORDER
↔NONE↔

BOUNDARY NODES IN INPUT NODE NUMBER ORDER

T 300= 6.7232

MODEL = SINDA85 FLUINT SAMPLE MODEL 5 - CPL GET AWAY SPECIAL (GAS) EXPERIMENT
FWDBCK

SUBMODEL NAME = EVAP

	CALCULATED			ALLOWED	
MAX DIFF DELTA T PER ITER	DRLXCC(EVAP	2000)=	0.	VS. DRLXCA=	1.000000E-02
MAX ARITH DELTA T PER ITER	ARLXCC(EVAP	1500)=	0.	VS. ARLXCA=	1.000000E-02
MAX DIFF DEL T PER TIME STEP	DTMPCC(EVAP	1000)=	6.494141E-02	VS. DTMPCA=	5.00000
MAX ARITH DEL T PER TIME STEP	ATMPCC(EVAP	1500)=	-1.403809E-03	VS. ATMPCA=	1.000000E+30
MIN STABILITY CRITERIA	CSGMIN(EVAP	1000)=	8.60554		
MAX STABILITY CRITERIA	CSGMAX(EVAP	2000)=	24.7744		
NUMBER OF ITERATIONS	LOOPCT	=	3	VS. NLOOPT=	100
PROBLEM TIME	TIMEN	=	30.0000	VS. TIMEND=	120.000
MEAN PROBLEM TIME	TIMEM	=	29.8616		
AVERAGE TIME STEP USED SINCE LAST OUTPUT		=	0.181818	VS. DTIMEI=	0.

DIFFUSION NODES IN INPUT NODE NUMBER ORDER

T 1000= 42.994 T 2000= 28.992

ARITHMETIC NODES IN INPUT NODE NUMBER ORDER

T 1500= 32.651

HEATER NODES IN INPUT NODE NUMBER ORDER
↔NONE↔

BOUNDARY NODES IN INPUT NODE NUMBER ORDER
↔NONE↔

MODEL = SINDA85 FLUINT SAMPLE MODEL 5 - CPL GET AWAY SPECIAL (GAS) EXPERIMENT
FWDBCK

FLUID SUBMODEL NAME = CPL ; FLUID NO. = 717

MAX TIME STEP = 161.798 ; LIMITING TANK = 20 REASON = TEMP./DENSITY CHANGE LIMIT
LAST TIME STEP = 0.811222 VS. DTMAXF/DTMINF = 10.0000 / 1.000000E-03; AVERAGE TIME STEP = 1.15138
PROBLEM TIME TIMEN = 120.000 VS. TIMEND = 120.000

LUMP PARAMETER TABULATION FOR SUBMODEL CPL

LUMP	TYPE	TEMP	PRESSURE	QUALITY	DENSITY	ENTHALPY	HEAT RATE	MASS RATE	ENERGY RATE
10	TANK	7.178	1.1323E+06	0.	628.8	2.1537E+05	0.	-2.7730E-08	5.2916E-02
20	TANK	7.048	1.1319E+06	0.	629.0	2.1477E+05	0.	-5.8717E-08	0.1092
40	TANK	29.48	1.1324E+06	1.000	8.760	1.4675E+06	-6.9385E-02	-2.9104E-10	-2.9345E-04
30	JUNC	29.01	1.1324E+06	1.000	8.782	1.4660E+06	0.	0.	0.
35	JUNC	29.53	1.1324E+06	1.000	8.757	1.4676E+06	0.3266	0.	3.8147E-06
50	JUNC	29.48	1.1324E+06	1.000	8.760	1.4675E+06	0.	-2.9104E-11	-6.1035E-05
1000	JUNC	29.00	1.1322E+06	1.000	8.781	1.4660E+06	254.5	0.	-8.5449E-04
1	JUNC	29.00	1.1323E+06	0.6617	13.17	1.0772E+06	-158.8	2.9104E-11	3.0518E-05
2	JUNC	29.00	1.1323E+06	0.3241	26.29	6.8919E+05	-157.9	0.	0.
3	JUNC	29.00	1.1323E+06	1.8600E-02	265.8	3.3811E+05	-142.8	2.9104E-11	0.
4	JUNC	16.70	1.1323E+06	0.	615.2	2.5932E+05	-32.05	0.	0.
5	JUNC	11.38	1.1323E+06	0.	622.9	2.3469E+05	-10.02	0.	0.
6	JUNC	8.727	1.1323E+06	0.	626.6	2.2249E+05	-4.963	0.	0.
7	JUNC	7.980	1.1323E+06	0.	627.7	2.1906E+05	-1.398	0.	0.
8	JUNC	7.394	1.1323E+06	0.	628.5	2.1636E+05	-1.096	0.	0.
9	JUNC	7.209	1.1323E+06	0.	628.7	2.1551E+05	-0.3451	0.	0.
100	JUNC	7.209	1.1323E+06	0.	628.7	2.1551E+05	0.	0.	0.
99	PLEN	29.00	1.1323E+06	0.1000	77.55	4.3165E+05	0.	1.4543E-07	3.1320E-02

MODEL = SINDA85 FLUINT SAMPLE MODEL 5 - CPL GET AWAY SPECIAL (GAS) EXPERIMENT
FWDBCK

FLUID SUBMODEL NAME = CPL ; FLUID NO. = 717

MAX TIME STEP = 161.798 ; LIMITING TANK = 20 REASON = TEMP./DENSITY CHANGE LIMIT
LAST TIME STEP = 0.811222 VS. DTMAXF/DTMINF = 10.0000 / 1.000000E-03; AVERAGE TIME STEP = 1.15138
PROBLEM TIME TIMEN = 120.000 VS. TIMEND = 120.000

TIE PARAMETER TABULATION FOR SUBMODEL CPL

TIE	TYPE	UA	QTIE	LUMP	TEMP.	NODE	TEMP.	PATH 1	FRACT	PATH 2	FRACT
35	RTN	9.2445E-02	0.3266	35	29.53	EVAP.1500	33.07	35	1.0000		
40	RTN	0.5299	-6.9385E-02	40	29.48	EVAP.2000	29.35	40	4.0000		
1000	RTM	15.96	254.5	1000	29.00	EVAP.1000	44.95				
1	RTNC	49.95	-158.8	1	29.00	PLATE.1	25.82	1	1.0000	2	0.5000
2	RTNC	36.55	-157.9	2	29.00	PLATE.2	24.68	2	0.5000	3	0.5000
3	RTNC	23.73	-142.8	3	29.00	PLATE.3	22.99	3	0.5000	4	0.5000
4	RTNC	6.517	-32.05	4	16.70	PLATE.4	11.79	4	0.5000	5	0.5000
5	RTNC	6.678	-10.02	5	11.38	PLATE.5	9.881	5	0.5000	6	0.5000
6	RTNC	6.758	-4.963	6	8.727	PLATE.6	7.996	6	0.5000	7	0.5000
7	RTNC	6.781	-1.398	7	7.980	PLATE.7	7.778	7	0.5000	8	0.5000
8	RTNC	6.798	-1.096	8	7.394	PLATE.8	7.237	8	0.5000	9	0.5000
9	RTNC	6.804	-0.3451	9	7.209	PLATE.9	7.163	9	0.5000	10	1.0000

MODEL = SINDA85 FLUINT SAMPLE MODEL 5 - CPL GET AWAY SPECIAL (GAS) EXPERIMENT
FWDBCK

FLUID SUBMODEL NAME = CPL ; FLUID NO. = 717

MAX TIME STEP = 161.798 ; LIMITING TANK = 20 REASON = TEMP./DENSITY CHANGE LIMIT
LAST TIME STEP = 0.811222 VS. DTMAXF/DTMINF = 10.0000 / 1.000000E-03; AVERAGE TIME STEP = 1.15138
PROBLEM TIME TIMEN = 120.000 VS. TIMEND = 120.000

PATH PARAMETER TABULATION FOR SUBMODEL CPL

PATH	TYPE	LMP 1	LMP 2	DUP I	DUP J	STAT	XL UPSTRM	FLOWRATE	DELTA PRES	FLOW REGIMES OR REYNOLDS NO.	
30	STUBE	30	35	45.	45.	NORM	1.000	4.5204E-06	3.607	408.37	
35	STUBE	35	40	1.0	2.0	NORM	1.000	2.0342E-04	0.9971	3791.4	
40	STUBE	40	50	1.0	1.0	NORM	1.000	4.0684E-04	0.9219	3550.1	
300	CAPIL	10	20	2.0	1.0	DLS	0.	2.0336E-04	355.3		
1000	NULL	20	1000	1.0	1.0	LS	0.	2.0342E-04	-209.0		
2000	NULL	1000	30	1.0	1.0	RLS	1.000	2.0342E-04	-209.0		
1	STUBE	50	1	1.0	1.0	NORM	1.000	4.0684E-04	10.62	3338.8	UNKNOWN
2	STUBE	1	2	1.0	1.0	NORM	0.6617	4.0684E-04	9.600	UNKNOWN	UNKNOWN
3	STUBE	2	3	1.0	1.0	NORM	0.3241	4.0684E-04	4.757	UNKNOWN	UNKNOWN
4	STUBE	3	4	1.0	1.0	NORM	1.8600E-02	4.0684E-04	4.814	UNKNOWN	219.47
5	STUBE	4	5	1.0	1.0	NORM	0.	4.0684E-04	4.168	219.47	206.83
6	STUBE	5	6	1.0	1.0	NORM	0.	4.0684E-04	4.244	206.83	200.81
7	STUBE	6	7	1.0	1.0	NORM	0.	4.0684E-04	4.382	200.81	199.15
8	STUBE	7	8	1.0	1.0	NORM	0.	4.0684E-04	4.336	199.15	197.85
9	STUBE	8	9	1.0	1.0	NORM	0.	4.0684E-04	4.423	197.85	197.44
10	STUBE	9	100	1.0	1.0	NORM	0.	4.0684E-04	2.178	197.44	197.44
100	STUBE	100	10	1.0	1.0	NORM	0.	4.0684E-04	3.745	197.44	197.37
99	STUBE	10	99	1.0	1.0	RLS	0.	1.4543E-07	5.7077E-03	7.05517E-02	8.98739E-02

MODEL = SINDA85
FWDBCK

FLUINT SAMPLE MODEL 5 - CPL GET AWAY SPECIAL (GAS) EXPERIMENT

SUBMODEL NAME = PLATE

		CALCULATED			ALLOWED	
MAX DIFF DELTA T PER ITER	DRLXCC(PLATE	3)=	9.094238E-03	VS. DRLXCA=	1.000000E-02
MAX ARITH DELTA T PER ITER	ARLXCC(	0)=	0.	VS. ARLXCA=	1.000000E-02
MAX DIFF DEL T PER TIME STEP	DTMPCC(PLATE	3)=	9.094238E-03	VS. DTMPCA=	1.000000E+30
MAX ARITH DEL T PER TIME STEP	ATMPCC(	0)=	0.	VS. ATMPCA=	1.000000E+30
MIN STABILITY CRITERIA	CSGMIN(PLATE	1)=	0.141930		
MAX STABILITY CRITERIA	CSGMAX(PLATE	19)=	7.60046		
NUMBER OF ITERATIONS	LOOPCT	=	1	VS. NLOOPT=	100
PROBLEM TIME	TIMEN	=	120.000	VS. TIMEND=	120.000
MEAN PROBLEM TIME	TIMEM	=	119.594		
AVERAGE TIME STEP USED SINCE LAST OUTPUT		=	0.857143	VS. DTIMEI=	0.

DIFFUSION NODES IN INPUT NODE NUMBER ORDER

T	1=	25.824	T	2=	24.684	T	3=	22.987	T	4=	11.789	T	5=	9.8805	T	6=	7.9965
T	7=	7.7784	T	8=	7.2370	T	9=	7.1628	T	11=	16.575	T	12=	15.509	T	13=	14.690
T	14=	9.9304	T	15=	9.3021	T	16=	7.7122	T	17=	7.7018	T	18=	7.1775	T	19=	7.1469

ARITHMETIC NODES IN INPUT NODE NUMBER ORDER
++NONE++

HEATER NODES IN INPUT NODE NUMBER ORDER
++NONE++

BOUNDARY NODES IN INPUT NODE NUMBER ORDER

T 300= 7.0936

MODEL = SINDA85
FWDBCK

FLUINT SAMPLE MODEL 5 - CPL GET AWAY SPECIAL (GAS) EXPERIMENT

SUBMODEL NAME = EVAP

		CALCULATED			ALLOWED	
MAX DIFF DELTA T PER ITER	DRLXCC(EVAP	2000)=	7.934570E-04	VS. DRLXCA=	1.000000E-02
MAX ARITH DELTA T PER ITER	ARLXCC(EVAP	1500)=	2.136230E-04	VS. ARLXCA=	1.000000E-02
MAX DIFF DEL T PER TIME STEP	DTMPCC(EVAP	2000)=	7.934570E-04	VS. DTMPCA=	5.00000
MAX ARITH DEL T PER TIME STEP	ATMPCC(EVAP	1500)=	2.136230E-04	VS. ATMPCA=	1.000000E+30
MIN STABILITY CRITERIA	CSGMIN(EVAP	1000)=	8.60554		
MAX STABILITY CRITERIA	CSGMAX(EVAP	2000)=	21.1310		
NUMBER OF ITERATIONS	LOOPCT	=	1	VS. NLOOPT=	100
PROBLEM TIME	TIMEN	=	120.000	VS. TIMEND=	120.000
MEAN PROBLEM TIME	TIMEM	=	119.594		
AVERAGE TIME STEP USED SINCE LAST OUTPUT		=	0.857143	VS. DTIMEI=	0.

DIFFUSION NODES IN INPUT NODE NUMBER ORDER

T	1000=	44.949	T	2000=	29.346												
T	1500=	33.065															

ARITHMETIC NODES IN INPUT NODE NUMBER ORDER

HEATER NODES IN INPUT NODE NUMBER ORDER
++NONE++

BOUNDARY NODES IN INPUT NODE NUMBER ORDER
++NONE++

MODEL = SINDA85 FLUINT SAMPLE MODEL 5 - CPL GET AWAY SPECIAL (GAS) EXPERIMENT
FWDBCK

FLUID SUBMODEL NAME = CPL ; FLUID NO. = 717

MAX TIME STEP (GROWTH LIMITED) = 1.98116 ; LAST CAUSE: TANK 40; FR CHANGE LIMIT IN PATH 35 (WAS 0.2476)
LAST TIME STEP = 7.324219E-04 VS. DTMAXF/DTMINF = 10.0000 / 1.000000E-02; AVERAGE TIME STEP = 0.289471
PROBLEM TIME TIMEN = 2520.00 VS. TIMEND = 2520.00

LUMP PARAMETER TABULATION FOR SUBMODEL CPL

LUMP	TYPE	TEMP	PRESSURE	QUALITY	DENSITY	ENTHALPY	HEAT RATE	MASS RATE	ENERGY RATE
10	TANK	17.57	1.1323E+06	0.	614.0	2.6337E+05	0.	-4.2102E-07	0.6926
20	TANK	17.18	1.1320E+06	0.	614.5	2.6156E+05	0.	-1.9424E-07	0.3313
40	TANK	29.58	1.1324E+06	1.000	8.755	1.4678E+06	5.9342E-02	2.0198E-08	2.9556E-02
30	JUNC	29.01	1.1324E+06	1.000	8.783	1.4660E+06	0.	-3.9290E-10	-5.4932E-04
35	JUNC	29.54	1.1324E+06	1.000	8.757	1.4676E+06	0.3418	0.	1.3351E-05
50	JUNC	29.58	1.1324E+06	1.000	8.755	1.4678E+06	0.	-1.9791E-09	-2.9297E-03
1000	JUNC	29.00	1.1322E+06	1.000	8.781	1.4660E+06	254.5	0.	-2.5482E-03
1	JUNC	29.00	1.1324E+06	0.8190	10.69	1.2579E+06	-88.68	-2.8813E-09	-3.6011E-03
2	JUNC	29.00	1.1324E+06	0.6439	13.53	1.0568E+06	-85.01	-3.6671E-09	-3.8910E-03
3	JUNC	29.00	1.1323E+06	0.4730	18.27	8.6039E+05	-83.00	-6.5193E-09	-5.6152E-03
4	JUNC	29.00	1.1323E+06	0.2977	28.52	6.5883E+05	-85.19	-3.3178E-09	-2.1667E-03
5	JUNC	29.00	1.1323E+06	0.1339	59.90	4.7059E+05	-79.56	-3.6671E-09	-1.7242E-03
6	JUNC	29.00	1.1323E+06	8.3778E-03	382.4	3.2636E+05	-60.96	0.	0.
7	JUNC	22.43	1.1323E+06	0.	606.9	2.8594E+05	-17.08	-6.9849E-09	-1.9989E-03
8	JUNC	18.99	1.1323E+06	0.	611.9	2.6996E+05	-6.753	-2.2701E-09	-6.1035E-04
9	JUNC	17.98	1.1323E+06	0.	613.4	2.6527E+05	-1.983	-1.1350E-09	-2.9755E-04
100	JUNC	17.98	1.1323E+06	0.	613.4	2.6527E+05	0.	0.	0.
99	PLEN	29.00	1.1323E+06	0.1000	77.55	4.3165E+05	0.	8.2251E-07	0.2166

MODEL = SINDA85 FLUINT SAMPLE MODEL 5 - CPL GET AWAY SPECIAL (GAS) EXPERIMENT
FWDBCK

FLUID SUBMODEL NAME = CPL ; FLUID NO. = 717

MAX TIME STEP (GROWTH LIMITED) = 1.98116 ; LAST CAUSE: TANK 40; FR CHANGE LIMIT IN PATH 35 (WAS 0.2476)
LAST TIME STEP = 7.324219E-04 VS. DTMAXF/DTMINF = 10.0000 / 1.000000E-02; AVERAGE TIME STEP = 0.289471
PROBLEM TIME TIMEN = 2520.00 VS. TIMEND = 2520.00

TIE PARAMETER TABULATION FOR SUBMODEL CPL

TIE	TYPE	UA	QTIE	LUMP	TEMP.	MODE	TEMP.	PATH 1	FRACT	PATH 2	FRACT
35	ETN	9.7409E-02	0.3418	35	29.54	EVAP.1500	33.05	35	1.0000		
40	ETN	0.5716	5.9342E-02	40	29.58	EVAP.2000	29.69	40	4.0000		
1000	ETM	15.96	254.5	1000	29.00	EVAP.1000	44.95				
1	ETNC	63.69	-88.68	1	29.00	PLATE.1	27.61	1	1.0000	2	0.5000
2	ETNC	49.36	-85.01	2	29.00	PLATE.2	27.28	2	0.5000	3	0.5000
3	ETNC	41.86	-83.00	3	29.00	PLATE.3	27.02	3	0.5000	4	0.5000
4	ETNC	35.91	-85.19	4	29.00	PLATE.4	26.63	4	0.5000	5	0.5000
5	ETNC	30.92	-79.56	5	29.00	PLATE.5	26.43	5	0.5000	6	0.5000
6	ETNC	11.93	-60.96	6	29.00	PLATE.6	23.89	6	0.5000	7	0.5000
7	ETNC	6.344	-17.08	7	22.43	PLATE.7	19.73	7	0.5000	8	0.5000
8	ETNC	6.448	-6.753	8	18.99	PLATE.8	17.95	8	0.5000	9	0.5000
9	ETNC	6.478	-1.983	9	17.98	PLATE.9	17.68	9	0.5000	10	1.0000

MODEL = SINDA85 FLUINT SAMPLE MODEL 5 - CPL GET AWAY SPECIAL (GAS) EXPERIMENT
FWDBCK

FLUID SUBMODEL NAME = CPL ; FLUID NO. = 717

MAX TIME STEP (GROWTH LIMITED) = 1.98116 ; LAST CAUSE: TANK 40; FR CHANGE LIMIT IN PATH 35 (WAS 0.2476)
LAST TIME STEP = 7.324219E-04 VS. DTMAXF/DTMINF = 10.0000 / 1.000000E-02; AVERAGE TIME STEP = 0.289471
PROBLEM TIME TIMEN = 2520.00 VS. TIMEND = 2520.00

PATH PARAMETER TABULATION FOR SUBMODEL CPL

PATH	TYPE	LMP 1	LMP 2	DUP I	DUP J	STAT	XL UPSTRM	FLOWRATE	DELTA PRES	FLOW REGIMES OR REYNOLDS NO.	
30	STUBE	30	35	45.	45.	NORM	1.000	4.6960E-06	3.789	424.23	
35	STUBE	35	40	1.0	2.0	NORM	1.000	2.1132E-04	1.064	3938.6	
40	STUBE	40	50	1.0	1.0	NORM	1.000	4.2262E-04	0.9932	3686.5	
300	CAPIL	10	20	2.0	1.0	DLS	0.	2.1113E-04	336.4		
1000	NULL	20	1000	1.0	1.0	LS	0.	2.1132E-04	-226.6		
2000	NULL	1000	30	1.0	1.0	RLS	1.000	2.1132E-04	-226.6		
1	STUBE	50	1	1.0	1.0	NORM	1.000	4.2262E-04	19.70	3467.0	UNKNOWN
2	STUBE	1	2	1.0	1.0	NORM	0.8190	4.2263E-04	23.86	UNKNOWN	UNKNOWN
3	STUBE	2	3	1.0	1.0	NORM	0.6439	4.2263E-04	18.00	UNKNOWN	UNKNOWN
4	STUBE	3	4	1.0	1.0	NORM	0.4730	4.2264E-04	12.62	UNKNOWN	UNKNOWN
5	STUBE	4	5	1.0	1.0	NORM	0.2977	4.2264E-04	12.26	UNKNOWN	UNKNOWN
6	STUBE	5	6	1.0	1.0	NORM	0.1339	4.2264E-04	6.293	UNKNOWN	UNKNOWN
7	STUBE	6	7	1.0	1.0	NORM	8.3778E-03	4.2264E-04	4.496	UNKNOWN	242.94
8	STUBE	7	8	1.0	1.0	NORM	0.	4.2265E-04	4.014	242.94	233.88
9	STUBE	8	9	1.0	1.0	NORM	0.	4.2265E-04	4.173	233.88	231.27
10	STUBE	9	100	1.0	1.0	NORM	0.	4.2265E-04	2.057	231.27	231.27
100	STUBE	100	10	1.0	1.0	NORM	0.	4.2265E-04	3.543	231.27	230.22
99	STUBE	10	99	1.0	1.0	RLS	0.	8.2251E-07	3.0729E-02	0.44662	0.50831

MODEL = SINDA85 FLUINT SAMPLE MODEL 5 - CPL GET AWAY SPECIAL (GAS) EXPERIMENT
FWDBCK

SUBMODEL NAME = PLATE

	CALCULATED			ALLOWED	
MAX DIFF DELTA T PER ITER	DRLXCC(PLATE	19)=	3.051758E-05	VS. DRLXCA=	1.000000E-02
MAX ARITH DELTA T PER ITER	ARLXCC(	0)=	0.	VS. ARLXCA=	1.000000E-02
MAX DIFF DEL T PER TIME STEP	DTMPCC(PLATE	19)=	3.051758E-05	VS. DTMPCA=	1.000000E+30
MAX ARITH DEL T PER TIME STEP	ATMPCC(	0)=	0.	VS. ATMPCA=	1.000000E+30
MIN STABILITY CRITERIA	CSGMIN(PLATE	1)=	0.117854		
MAX STABILITY CRITERIA	CSGMAX(PLATE	19)=	7.60046		
NUMBER OF ITERATIONS	LOOPCT	=	1	VS. NLOOPT=	100
PROBLEM TIME	TIMEN	=	2520.00	VS. TIMEND=	2520.00
MEAN PROBLEM TIME	TIMEM	=	2520.00		
AVERAGE TIME STEP USED SINCE LAST OUTPUT		=	0.784314	VS. DTIMEI=	0.

DIFFUSION NODES IN INPUT NODE NUMBER ORDER

T	1=	27.611	T	2=	27.281	T	3=	27.020	T	4=	26.630	T	5=	26.428	T	6=	23.892
T	7=	19.733	T	8=	17.946	T	9=	17.678	T	11=	22.443	T	12=	22.338	T	13=	22.194
T	14=	21.677	T	15=	21.804	T	16=	20.361	T	17=	18.744	T	18=	17.557	T	19=	17.566

ARITHMETIC NODES IN INPUT NODE NUMBER ORDER
↔NONE↔

HEATER NODES IN INPUT NODE NUMBER ORDER
↔NONE↔

BOUNDARY NODES IN INPUT NODE NUMBER ORDER

T 300= 17.000

MODEL = SINDA85 FLUINT SAMPLE MODEL 5 - CPL GET AWAY SPECIAL (GAS) EXPERIMENT
FWDBCK

SUBMODEL NAME = EVAP

	CALCULATED			ALLOWED	
MAX DIFF DELTA T PER ITER	DRLXCC(EVAP	2000)=	3.051758E-05 VS.	DRLXCA=	1.000000E-02
MAX ARITH DELTA T PER ITER	ARLXCC(EVAP	1500)=	-6.103516E-05 VS.	ARLXCA=	1.000000E-02
MAX DIFF DEL T PER TIME STEP	DTMPCC(EVAP	2000)=	3.051758E-05 VS.	DTMPCA=	5.00000
MAX ARITH DEL T PER TIME STEP	ATMPCC(EVAP	1500)=	-3.051758E-05 VS.	ATMPCA=	1.000000E+30
MIN STABILITY CRITERIA	CSGMIN(EVAP	1000)=	8.60554		
MAX STABILITY CRITERIA	CSGMAX(EVAP	2000)=	19.9581		
NUMBER OF ITERATIONS	LOOPCT	=	1 VS.	NLOOPT=	100
PROBLEM TIME	TIMEN	=	2520.00 VS.	TIMEND=	2520.00
MEAN PROBLEM TIME	TIMEM	=	2520.00		
AVERAGE TIME STEP USED SINCE LAST OUTPUT		=	0.784314 VS.	DTIMEI=	0.

DIFFUSION NODES IN INPUT NODE NUMBER ORDER

T 1000= 44.951 T 2000= 29.688

ARITHMETIC NODES IN INPUT NODE NUMBER ORDER

T 1500= 33.047

HEATER NODES IN INPUT NODE NUMBER ORDER
++NONE++

BOUNDARY NODES IN INPUT NODE NUMBER ORDER
++NONE++

PROBLEM F: LH_2 STORAGE TANK

This problem simulates a liquid hydrogen storage tank with a thermodynamic vent system (TVS), and includes both steady-state TVS sizing and transient filling. This problem was chosen to demonstrate the following FLUINT features:

1. nonequilibrium tanks (imperfect mixing)
2. 7000 series fluid descriptions
3. user-supplied heat transfer coefficients
4. steady-state sizing logic/models vs. transient simulation logic/models
5. multiple ties to one lump and to one node
6. use of heater junctions to simplify heat exchanger simulations
7. moveable ties

F.1 Problem Description

To minimize resupply launches and to maximize spacecraft life, long term storage of cryogenic propellants, sensor coolants, and fuels in space requires absolute minimal boil-off. With this requirement in mind, a liquid hydrogen ground test system (Fig. F–1) was built and tested as part of a Martin Marietta project headed by Dr. John Anderson. The test system consists of a four foot diameter storage tank that is blanketed by multi-layer insulation (MLI), and uses a TVS to minimize losses. The tank is suspended from above using low conductivity filaments. Liquid is drained from the bottom of the tank, flashed through a pressure regulator, and passed back through a heat exchanger that is completely submerged in the liquid that is inside of the tank. Within this internal heat exchanger, the remaining liquid (now at significantly reduced pressure and several degrees colder) is boiled, absorbing energy from the tank. The heat exchanger discharges vapor to a vapor cooled shield (VCS), which is a thin aluminum shell located within the MLI. Aluminum piping is attached to the VCS shell in an upward spiral, with approximately constant spacing between coils. This vapor passing through this pipe cools the shield, reducing heat leaks into the system. The warmed vapor hydrogen is then vented to vacuum through a control valve.

The goals of this analysis are to (1) calculate the performance of a half-full tank at the optimum TVS flowrate, and (2) simulate the transient filling of the tank from a colder, higher pressure source.

The parameters of the system are:

Tank:

Inner diameter	42 in
Volume	22.45 ft3
Thickness	0.3 in
Conductance at top from environment	1.24E-2 BTU/°F
Material	Steel
Conductivity	8.33 BTU/hr-ft-°F
Specific Heat	0.11 BTU/lb_m-°F

* John E. Anderson et al, *Evaluation of Long-term Cryogenic Storage System*, Cryogenic Engineering Conference, July 1989.

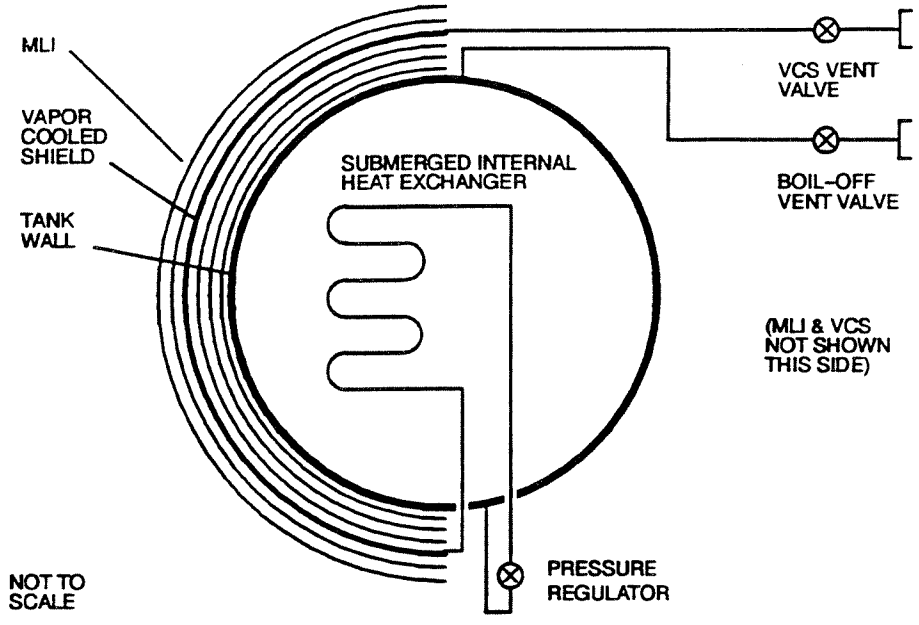


Figure F-1 Schematic of Storage Tank with Thermodynamic Vent System

Submerged Internal Heat Exchanger:

Piping inner diameter	0.18 in
T-shaped Fin . , each of 3 sections	2in x 1/16in
Total pipe length	160 in.
Material	Copper
Conductivity	500. BTU/hr-ft-°F

VCS:

Thickness	0.045 in
Conductance at top from environment	1.86E-4 BTU/°F
Material	Aluminum
Conductivity	133. BTU/hr-ft-°F
Specific Heat	0.21 BTU/lb_m-°F

VCS Piping:

Diameter	0.313 in
Thickness	0.035 in
Total Length . (Spiral)	80. ft
Material	Aluminum
Conductivity	133. BTU/hr-ft-°F

MLI:

Outer Diameter	51 in
Tank to VCS effective emissivity (ε^*)	0.008
VCS to skin effective emissivity (ε^*)	0.007
Skin emissivity	0.1
Chamber temperature	540 R

Internal tank heat transfer:

Liquid to subcooled walls	20 BTU/hr-ft^2-°F
Liquid to superheated walls	Pool boiling (Sample Problem C)
Vapor to subcooled walls	20 BTU/hr-ft^2-°F (condensing)
Vapor to superheated walls	4 BTU/hr-ft_2-°F

TVS:

Flowrate (initial)	0.0382 lb_m/hr
Nominal pressure	4 psia

Fill Source:

Temperature	Minimum (near freezing)
Pressure	60 psia

Some of the above parameters were calculated by analyses that are not documented in this section, but that merit some discussion.

The MLI performance was estimated from test data—effective MLI emissivities are notoriously difficult to predict *a priori*, and their prediction is not relevant to this problem in any case. To calibrate these factors, data from a boil-off (vented vapor, constant pressure) test were used. Only two values are needed from such a test: the VCS temperature and the rate of boil-off. Data from other tests with zero and excess TVS flowrate provided an independent confirmation of these values and of the model in general.

One of the primary purposes of such a model is to help predict the required TVS flowrate, which is a significant inverse figure of merit for such a storage system: a lower value indicates better insulation and less valuable liquid lost. Because of the long time scales involved in such testing (it takes a full week to reach equilibrium), it is extremely difficult to determine the minimum (and therefore optimum) TVS flowrate required to keep the tank pressure from rising. By the time such tests have been completed, the LH_2 fill level has changed so drastically (perhaps the tank has drained completely) that the results may be indeterminate—a steady value is required from an inherently unsteady system. Analytically, arriving at a design value for the TVS flowrate is much more tenable: the TVS flowrate is equal to the heat rate that enters the tank divided by the heat of vaporization. Unfortunately, this heat rate and the TVS flowrate are very tightly coupled.

While complicated iterative logic could be created to guide a model toward the design flowrate in the course of a FASTIC or STDSTL run, the tight coupling noted above would make convergence very elusive. It was deemed simpler to perform a parametric set of steady-state runs of system response versus TVS flowrate, the results of which are depicted in Figure F–2. This method predicts a TVS flowrate of about 0.0382 lb_m/hr corresponding to a heat leak into the tank of about 7 BTU/hr. As the bonus of using this method, the system response to off-design conditions can also be seen in this same chart. For example, it is evident from this figure that at zero TVS flowrate the heat leak into the tank would be almost five times greater than at the design flowrate, with a proportional increase in the amount of liquid lost due to boil-off: there is a substantial benefit derived from the incorporation of the TVS.

F.2 A FLUINT Model

The modeling approach is governed by the goals of the analysis: to capture the behavior of the whole system, not to provide detailed point-design analyses of its constituent components. The tank wall, VCS, and MLI were each divided equally by area into nine horizontal sections. Figure F–3 shows the schematic of the mathematical model.

The decision to divide the model into nine sections was driven by the need to resolve the heat transfer (1) within the VCS and (2) between the tank wall and its fluid with adequate spatial resolution. Actually, because of the relatively large conductivities in the VCS and tank walls, fewer sections would probably suffice—the system can be characterized by fewer temperatures.

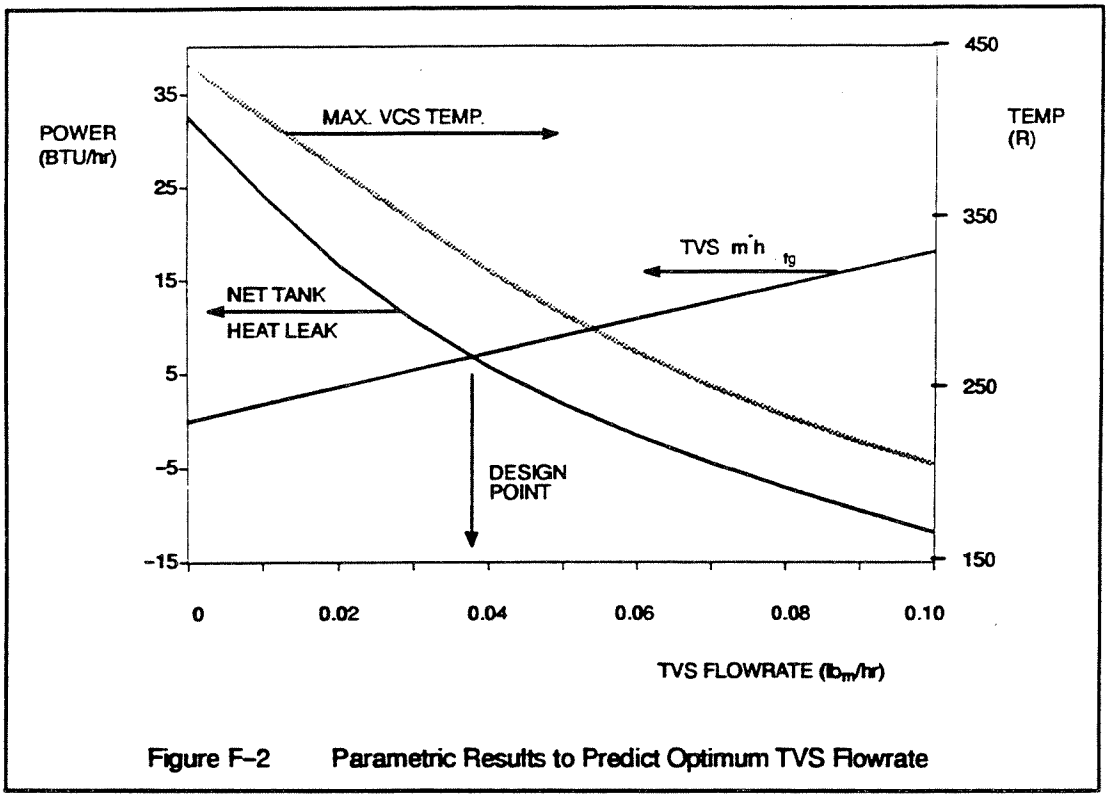


Figure F-2 Parametric Results to Predict Optimum TVS Flowrate

Because of the large temperature variations in the aluminum VCS, temperature-varying conductivities and specific heats might be appropriate, and are easily included if the property information is readily available. However, these quantities were assumed constant in this analysis for simplicity. While each node in the tank wall has the same mass, the conductivities between them vary according to the spherical geometry. The same is true for the VCS shell nodes. These variations are estimated based on rough KA/L style calculations.

A single FLUINT control volume (or "tank") could be used to simulate the storage tank itself. This decision would precludes thermal stratification, which is valid because of the relatively large conductivity of the thick tank wall and of the surrounding VCS. However, the use of a single tank would imply perfect mixing, which would lead to sudden collapse of the vapor volume when the fill transient began. The fluid is not well stirred: nonequilibrium routines NONEQ0 and NONEQ1 will be used, and the tank will be divided into two FLUINT tanks, one for the vapor phase and another for the liquid. As required by those routines, a NULL connector will be introduced between them. (The use of a twinned NULL connector is normally recommended between NONEQ tanks since it expedites simulations where condensation happens continuously in the vapor tank or where boiling happens continuously in the liquid tank. However, no significant differences in execution speed were noted in this specific example.)

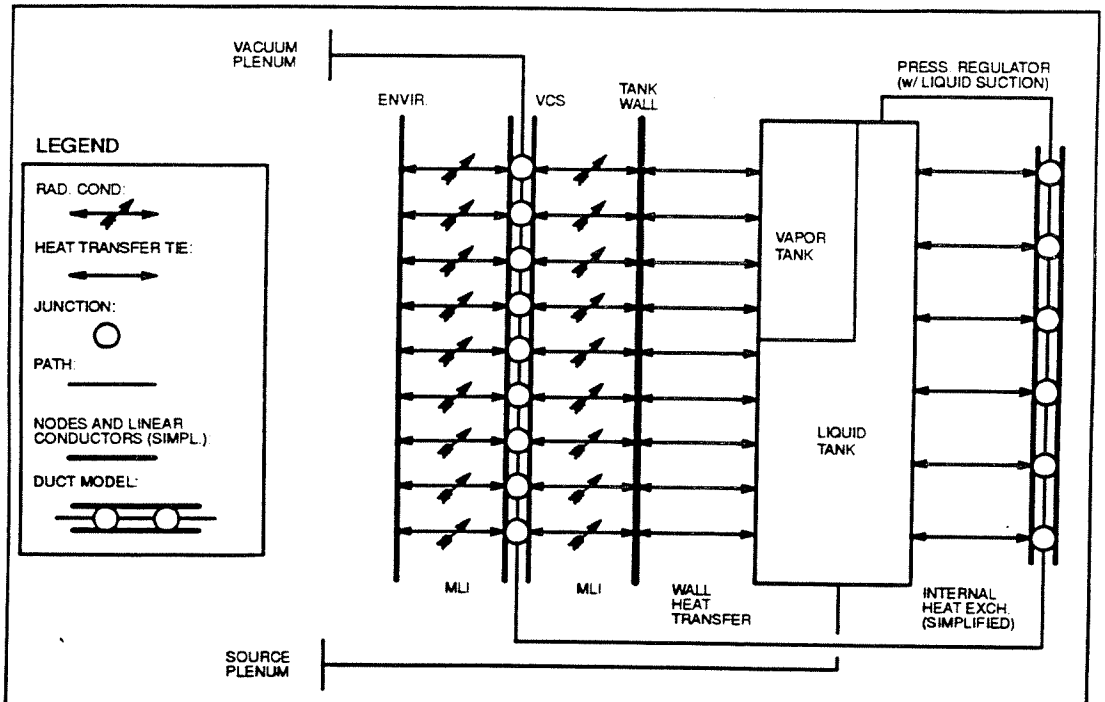


Figure F-3 Schematic of Storage Tank Math Model

The heat transfer between the phases is difficult to predict with any certainty, but can be assumed to be less than the heat transfer to the wall due to the lack of shear at the interface. The NONEQ1 default value multipliers were used to input these coefficients, which are influenced by the presence of natural convection in the vapor side and the stirring effect of injected fluid in the liquid side. The reader is welcome to investigate the transient fill problem by parametrically varying these values. The interface area *is* known with a high degree of certainty, and is updated in the logic blocks accordingly.

The heat transfer between the tank wall and the fluid within is dependent not only on the phase of the adjacent fluid, but also on effective "state" of the wall (i.e., is it cold enough to cause local condensation, or warm enough to cause local boiling?). At different fill levels, the fixed wall nodes will be adjacent to a different FLUINT tank. This difficulty is overcome by the use of the PUTTIE routine, which allows the ties to be reattached as the tank fills or empties.

For this system, there is no true time-independent state unless the insulation is perfect: the tank must eventually empty. However, the fill level varies slowly (by definition—otherwise the storage system is a failure). A pseudo-steady solution may be found by using HLDLMP on both tanks during steady-state solutions, which is equivalent to treating them like plena. The tanks will be subsequently "released" to simulate transient filling.

A path with liquid suction is used to extract only liquid from the liquid tank, which would otherwise be assumed to be homogeneously distributed. While two-phase flow *can* occur within such a divided-phase (nonequilibrium) model, the importance of the liquid suction is minimal unless a single mixed tank were used. The effect of gravity (i.e., fill level) on this pipe penetration is minimal because of the low density of hydrogen.

This same path, an STUBE, will be used to simulate an idealized pressure regulator. The pressure regulator could be modeled as a simple LOSS element, but the K-factor is unknown and must be sized during the steady-state run such that the TVS line is held at 4 psia. A simple way to achieve this sizing without adjusting the K-factor dynamically is to use an STUBE. Before steady-state, an HC coefficient will be calculated once to achieve this sizing. Before the transient, the HC can be zeroed and the STUBE FK factor updated to supply to necessary pressure drop that was formerly imposed by the HC constant.

An analogous problem exists at the exhaust to "vacuum." An MFRSET is ideal for the steady-state run, but a LOSS is desired during the transient. In this case, the paths are switched using DUP factors, and the FK for the LOSS element is sized from the MFRSET-caused pressure drop.

Because of the relative sizes and time scales, nearly all of the fluid model employed time-independent junctions and connectors, while the thermal model included capacitance lags of the tank and VCS. In other words, the hydrodynamic response was assumed instantaneous compared to the thermal response.

The liquid tank is tied via HTU ties to the fins of the internal heat exchanger. Initial models showed that this internal heat exchanger is greatly oversized because of the small flowrate needed to balance the small heat leak into the tank. The inlet fluid, initially at a quality of about 0.18, boils quickly in the first few inches of the twelve foot long heat exchanger. Furthermore, for all flowrates of interest, the vapor exits at the temperature of the liquid in the tank—the effectiveness of the heat exchanger is unity. To capture the boiling would require a very fine mesh network at the inlet. Such detail is really only needed to help design the heat exchanger itself, and is unimportant to the operation of the system. Therefore, a modeling "trick" is employed to greatly simplify the heat exchanger model without sacrificing its hydrodynamic or thermodynamic behavior. The trick is to use a "heater junction" (via HTRLMP) in the first leg of the heat exchanger to decouple the thermodynamic solution without disrupting the hydrodynamic solution: the heater junction is used to convert the flow into vapor without considering the heat transfer process involved. Energy is conserved because the energy required to perform this evaporation is extracted from the tied node. Because of the complexity of the internal heat exchanger design and its lack of relative importance, further discussions of this portion of the model are not necessary.

The description of the working fluid itself deserves attention. FLUINT contains a "library" of properties for 20 room temperature refrigerants. To analyze a fluid such as hydrogen, the user must provide a description in the input file using an FPROP block. In this example, a simplified two-phase (7000 series) description of hydrogen was developed using handbook values for the properties of each phase and for the properties of the saturation dome. From these values, FLUINT develops a thermodynamically consistent internal description of hydrogen. Tabular descriptions (input as 6000 series fluids) also exist, and would normally represent a better choice because they are faster, more accurate, and valid over a wide range than a 7000 series fluid.

F.3 The Input File

The input file is listed immediately following the last section.

OPTIONS DATA—As with previous examples, a user file (USER1) is named that will contain a summary listing of the output file. A restart file is used to save the results of the steady-state analysis for future use.

CONTROL DATA—English units are chosen, with units of °R and psia. 500 steady-state iterations are allowed, and the EXTRAP accelerator (governed by EXTLIM and ITERXT) is constrained. The value of OUTPTF is given as 1 minute versus 6 minutes for the value of OUTPUT; the fluid submodel OUTPUT CALLS block is used for frequent printing the USER1 file, while the thermal submodel OUTPUT CALLS block is used for standard output calls.

USER DATA—Variables to be used in the logic are declared and initialized. Among them are heat transfer coefficients, and various geometric properties. NTEST is used as both a restart flag and as a storage cell for the restart record number.

NODE DATA—Four thermal submodels are input: (1) *IHX*, the internal heat exchanger consisting of 6 arithmetic nodes along the copper pipe and fin (#1-6) and four more nodes at other locations lacking a pipe, (2) *TANK*, the model of the nine vertical sections of steel wall, (3) *VCS*, a similar model for the aluminum VCS shell, and (4) *MLI*, a corresponding model of the outer layer of MLI, which is assumed massless. The MLI model also contains the boundary node representing the chamber source.

CONDUCTOR DATA—For the IHX model, conductors are input for the axial terms. In the VCS and TANK models, linear conductors are used for vertical conduction with varying KA/L, where A and L are only roughly approximated. The MLI submodel contains the effective radiation conductances linking the tank, the VCS, the MLI, and the chamber. These conductors assume nearly planar radiation between the vertical segments as a first order approximation.

FLOW DATA—One fluid submodel named HYDRO is used. The working fluid is "7702", meaning the simplified user-input two-phase parahydrogen description discussed below. The names and types of the major elements are as follows:

Tank model:
- TANK 1 ..... liquid side of tank
- TANK 2 ..... vapor side of tank
- NULL 1 ..... between tanks for NONEQ calls
- HTU ties 201-255 ..... ties to *outside* of internal heat exchanger
- HTU ties 301-309 ..... ties to tank wall

Throttle and internal heat exchanger (IHX) model:
- STUBE 100 ..... pressure regulator (throttle)
- HTN ties 101-106 ..... ties to *inside* of internal heat exchanger
- STUBEs 101-106 ..... IHX line
- JUNCs 101-106 ..... IHX line

VCS model:
- HTNC ties 401-409 ..... ties to VCS plate with fin efficiency term
- STUBEs 401-410 ..... VCS line
- JUNCs 401-409 ..... VCS line

Flow boundary conditions:

MFRSET 998	outlet valve to "vacuum"
LOSS 999	replaces above MFRSET during transient
PLEN 999	"vacuum" (1 psia, warm vapor)
PLEN 1999	fill source (60 psia, cold liquid)
STUBE 1999	fill line (invisible during steady state)

ARRAY DATA—Arrays of void fractions versus segment position are input. These arrays are used in the FLOGIC 0 block to locate the liquid surface with respect to the wall nodes.

FLOGIC 0—This routine sets up the nonequilibrium model (NONEQ0), updates the somewhat detailed model of the tank wall heat transfer. The heat transfer model adjusts the 300 series ties according to fill level and the temperature of the wall compared to the saturation point. The logic loops through these ties from bottom to top, locating the position of the interface compared to the wall segments, which have equal heat transfer area but unequal depths due to the spherical geometry of the storage tank.

If the tank wall is at least partially covered and is warmer than the saturation temperature, then pool boiling heat transfer is included by a call to PBOIL, a routine supplied within SUBROUTINE DATA that was documented in the pressure cooker sample problem. For colder submerged walls, a single-phase conduction value is assumed. Similarly, for vapor segments the heat transfer coefficient is a function of whether the wall temperature is above or below saturation. If the wall temperature is below saturation, an estimated condensation coefficient is used. Note that as the liquid level rises or recedes, a single wall node will "see" a different tank. This change is effected via the PUTTIE routine, which can also be used to tie to other nodes or to move ties completely.

FLOGIC 1—This logic block completes the calls to the nonequilibrium model by calling NONEQ1. Before this call, the interface area is calculated as a function of fill level, which is related to the size of the vapor volume (Tank #2). An internal FLUINT routine is used to calculate a cube root; to prevent the debug feature from flagging out the name "CUROOT" as an unidentified variable, an 'F' is placed in column 1 to prevent attempted translation and therefore validation. Alternatively, the debug feature may be turned off locally with a DEBOFF command, or globally with a NODEBUG command in OPTIONS DATA.

Although the surface area is well characterized, the interfacial heat transfer coefficients can only be estimated. These coefficients are lower than those to the wall (user constants HTVAP and HTLIQ) because of the lack of slip at the interface. For this example, values of ULIQ and UVAP are input as -4.0 and -2.0, meaning four times the liquid solid conduction value and two times the vapor conduction value, both of which are dependent only on the shape of the volumes. Values higher than the lower limit are warranted because of natural convection in the vapor and ingress in the liquid, which causes stirring. Both of these phenomena diminish as the tank fills and the inlet flowrate drops, but the multipliers are left constant due to lack of better information. As with any heat transfer coefficient, these assumptions are open to criticism; the reader is encouraged to try parameterizations.

OUTPUT CALLS—The infrequently called block for the MLI submodel contained calls to the standard FLUINT tabulation routines as well as TPRINT. During transients, the block for HYDRO will be called more frequently, printing a single line of tailored information to the user file with each call.

OPERATIONS DATA—The configuration is built, containing all input submodels. If NTEST has been set to read a restart file, then the logic jumps below to the start of the transient logic. Otherwise, the two tanks are held (e.g., put in a boundary state) by HLDLMP for FASTIC, and will be released before the transient via calls to RELLMP. Similarly, the inlet of the internal heat exchanger, Junction #101, is turned into a saturated vapor heater junction by calls to CHGLMP and to HTRLMP, which causes the enthalpy to become fixed. The energy required to boil all of the liquid (entering at a quality of about 0.18) in #101 is automatically extracted from the tied node IHX.1. This represents heat transfer by a nonphysical process, but prevents the program from requiring finer spatial resolution to discern the boiling process, which occurs within a few inches of pipe. Because the remainder of the heat exchanger brings the TVS flow up to the temperature of the liquid temperature by the exit, the entire heat exchanger could be replaced by one such tie to a HTRLMP junction—a simple representation of a perfect heat exchanger.

The pressure regulator (Joule-Thompson throttle) is designed to assure 4 psia at the outlet. It could be modeled by a LOSS whose FK is continually adjusted to achieve such performance. A simpler and more stable method is to use the HC coefficient of an STUBE to place a prescribed pressure drop across the isenthalpic device. For demonstration purposes, the STUBE is later converted into a LOSS-like connector using its simulation parameters IPDC, FC, and FPOW. (Using the FK factor would normally be an easier choice.) When IPDC = 0, the user assumes control of the FC and FPOW parameters, and the values of TLEN, WRF, and UPF become irrelevant. FPOW=1.0 corresponds to a loss that scales with the square of velocity (i.e., a K-factor loss); FPOW=0.0 corresponds to a linear (laminar) loss.

To model the exhaust valve, an MFRSET is used during steady-states to simplify the model (again, this is easier than trying to dynamically size a LOSS factor. Replacing the MFRSET with an equivalent LOSS allows the TVS flowrate to vary with upstream tank pressure during the transient. DUP options are used to turn off one path and turn on another. Recall that a path with DUPI=DUPJ=0 still exists, it is simply ignored by the adjacent lumps. The lumps will affect the path, but not vice versa.

DUP factors are similarly exploited to make the fill line (STUBE 1999) "appear" suddenly at the start of a transient. A CTLVLV would accomplish the same, but would require an additional junction if it did not replace the STUBE completely, and would add a small pressure drop which might be undesirable.

FPROP DATA—The description of the fluid itself deserves some discussion. Two-phase parahydrogen may be adequately described by a simple 7000 series fluid as long as the vapor pressures never exceed a fraction of the critical point. This particular description contains arrays instead of single values and point-slope properties. It is very accurate along the isobar representing one atmosphere, and is reasonably accurate for slightly higher pressures. The second DOME input line could be replaced by the commented line for even greater accuracy within the tank itself, but the accuracy at lower pressures (e.g., within the TVS) would suffer slightly.

F.4 Output Description

The user summary file and the selected portions of the OUTPUT file are listed following the input file. Figure F–4 contains a plot detailing the fill event.

Because the liquid entering the tank is colder than the initial tank state, a perfect mixing assumption would cause an immediate collapse of the tank pressure (as happens in the pressure cooker sample problem), and this lowered pressure would persist until the liquid tank became hard-filled within a few minutes, at which time the pressure in the tank would suddenly jump up to the source pressure of 60 psia. Modeling the storage tank with a single FLUINT tank will result in such behavior.

With interphase heat and mass transfer reduced to finite time scales, the cold fluid entering the tank compresses the vapor, causing an increase rather than a decrease in tank pressure. Even for very large heat transfer coefficients (i.e., nearer to perfect mixing), the initial influx of liquid is so swift compared to the condensation rate that the vapor compresses rather than collapses. The influx thus leads a decrease in the flowrate: a negative feedback loop exists. Because the liquid is colder, the vapor will begin to condense. In fact, the condensation rate in this particular case is fast enough to cause wall condensation and/or fogging (or, in this case with a gravity gradient, raining) in the bulk vapor, but is not fast enough to cause a collapse. If the condensation rate had been more rapid, the NULL connector could have been twinned to yield speed savings by accounting for the rain (with one connector) as well as the condensation on the surface (with its twin). The only change to the input file would have been the addition of a "TWIN=1001" to the definition of the NULL connector.

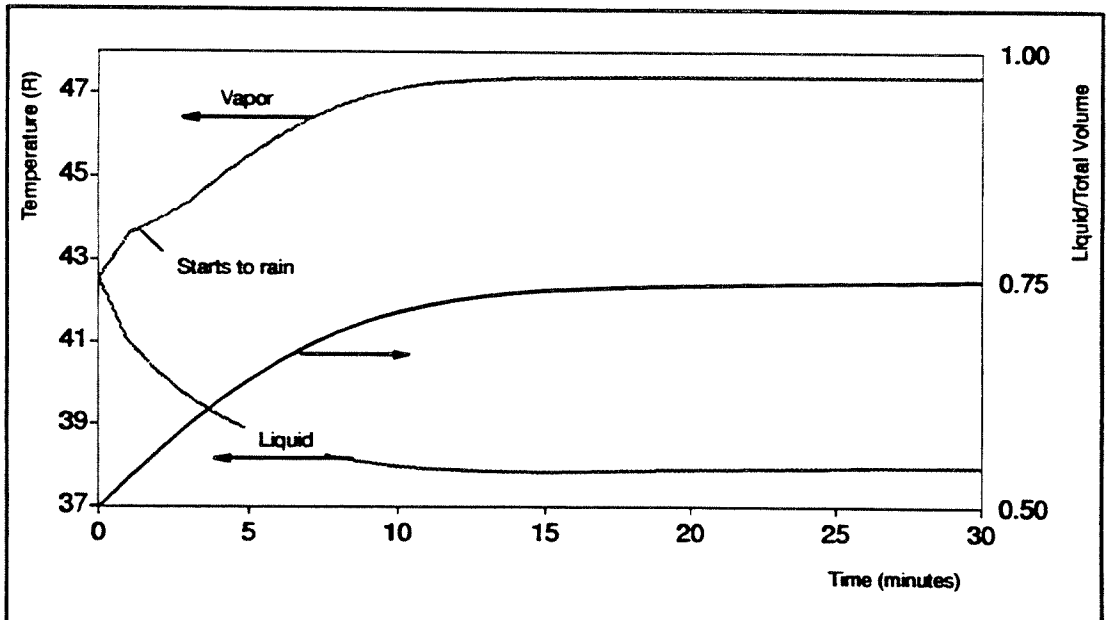


Figure F–4 Temperature Response During Fill

After about 15 minutes, the system is quasi-steady even though the tank is not full: the pressures have equilibrated and the influx rate is matched by the condensation rate within the vapor. Liquid is added to Tank #1 in the form of warm condensate at the top, and the resulting decrease in vapor volume is offset by further addition of cold liquid at the bottom. Because the liquid tank gains more cold liquid than warm condensate, the temperature would continue to drop if no wall model were included. In this analysis, the liquid eventually starts to warm because of energy absorbed from the environment. The TVS flowrate has increased due to the increased tank pressure, but either this increase is negligible or, more likely, the VCS temperature is decreasing very slowly and thus has not intercepted much additional energy. (The more pressure in the tank, the *less* equilibrium TVS flowrate is required to maintain that saturation condition. However, such a system would be unstable: a transient increase in pressure needs to be offset by a transient *increase* in TVS flowrate.)

Input File

```
HEADER OPTIONS DATA
TITLE LH2 STORAGE TANK WITH INTERNAL HX AND VAPOR COOLED SHIELD
         MLINE    = 52
         MODEL    = TVS
         OUTPUT   = tvs.out
         RSO      = tvs.rso
         RSI      = tvs.rsi
         USER1    = tvs.usr
C        USER2    = tvs.us2
C
HEADER CONTROL DATA, GLOBAL
         UID      = ENG              $ ENGLISH UNITS
         ABSZRO   = 0.0              $ DEGREES RANKINE
         PATMOS   = 0.0              $ PSIA
         NLOOPS   = 500
         SIGMA    = .1712E-8         $ S-B CONSTANT IN ENGLISH UNITS
         EXTLIM   = 5.0
         ITERXT   = 6
         OUTPUT   = 0.1
         OUTPTF   = 1.0/60.0         $ OUTPUT EACH MINUTE
C
HEADER USER DATA,GLOBAL
         HTLIQ    = 20.0             $ TANK TO BULK LIQ HTC
         HTVAP    = 4.0              $ TANK TO BULK VAP HTC, SINGLE PHASE
         HTCON    = 20.0             $ TANK TO BULK VAP HTC, CONDENSATION
C ABOVE MAY SEEM LOW BUT DOMINATED BY HTVAP: MUST GET TO WALL FIRST.
         TSAT     = 1.0              $ SAT TEMP
         ASUR     = 1.0              $ INTERFACE SURFACE AREA
         VTANK    = 22.45            $ TOTAL TANK VOLUME
         RTANK    = 1.75             $ TANK RADIUS (FT)
         AWALL    = 38.5/9.          $ AREA OF WALL, EACH SEGMENT
         PI       = 3.1416
         NTEST    = 0                $ RESTART RECORD NUMBER
HEADER NODE DATA,IHX
C INTERNAL HEAT EXCHANGER: ALL ARITHMETIC NODES; NEGLECT CAPACITANCE OF
C 1/16" COPPER FINS. (DOWNSTREAM DISCRETIZED IN FLUINT MODEL)
C
C ALMOST ALL BOILING HAPPENS QUICKLY IN FIRST FOOT
C TOTAL LENGTH IS 160" WITH ELBOWS EVERY APPROX 27"
C DIVIDE INTO 6 SEGMENTS 2.22'  EACH
C
         GEN 1,6,1,43.,-1.0         $ PIPE/FIN NODES (@ ROOT OF FINS)
         201,43.,-1.0               $ MIDDLE OF UNPIPED SECTIONS
         202,43.,-1.0               $ MIDDLE OF UNPIPED SECTIONS
         204,43.,-1.0               $ MIDDLE OF UNPIPED SECTIONS
         205,43.,-1.0               $ MIDDLE OF UNPIPED SECTIONS
```

```
HEADER CONDUCTOR DATA,IHX
C INTERNAL HEAT EXCHANGER: AXIAL CONDUCTION TERMS
C                          pipe A    fin A     K  L
         GEN 1,5,1,1,1,2,1,1.64E-4 +1.74E-3 *500./2.22 $ AXIAL
         100,1,TANK.1,1.64E-4 *500./1.0 $ TO TANK BASE (SAY 1 FT PIPE)
C ACTUALLY, #6 AND TANK.1 ARE NEARLY CO-LOCATED
         101,6,TANK.1,1.64E-4*500.*12.0 $ TO TANK BASE (SAY 1 IN PIPE)
         201,1,201,1.74E-3 *500./1.25        $ TO UNPIPED SECTIONS
         202,2,202,1.74E-3 *500./1.25        $ TO UNPIPED SECTIONS
         204,4,204,1.74E-3 *500./1.25        $ TO UNPIPED SECTIONS
         205,5,205,1.74E-3 *500./1.25        $ TO UNPIPED SECTIONS
         203,201,202,1.74E-3 *500./2.5       $ TWEEN UNPIPED NODES
         206,204,205,1.74E-3 *500./2.5       $ TWEEN UNPIPED NODES
C
C OUTER SHELL MLI PLUS ALL MLI CONDUCTANCES AND CHAMBER BDY NODE
C
HEADER NODE DATA,MLI
         GEN 1,9,1,530.,-1.0         $ FROM BASE UP
         -999,540.,0.0               $ VACUUM CHAMBER BDY NODE
C
HEADER CONDUCTOR DATA,MLI
C CONDUCTORS BETWEEN CHAMBER AND OUTER LAYER
C SAY 51" DIA @ 0.04 e, CHAMBER 72" DIA @ 0.3 e: eff e = 0.034
C BUT KNOW FROM TESTS THAT EXCHANGE IS GOOD HERE: SAY e = 0.1
         GEN,-1,9,1,1,1,999,0,56.7/9.*0.1
C CONDUCTORS BETWEEN OUTER LAYER AND VCS (24 LAYERS MLI) say e* = .007
         GEN,-11,9,1,VCS.1,1,1,1,50.3/9.*.007
C CONDUCTORS BETWEEN VCS AND TANK (48 LAYERS MLI) say e* = .008
         GEN,-21,9,1,VCS.1,1,TANK.1,1,50.3/9.*.008
C
C NOW LINEAR CONDUCTORS TO VCS AND TO TANK:
C (ALL AT TOP OF TANK AND VCS FOR SIMPLICITY)
C
         100,999,TANK.9,1.24E-2
         200,999, VCS.9,1.86E-4
```

```
HEADER NODE DATA,TANK
C TANK WALL NODES: DIVIDE TANK INTO 9 SECTIONS REPRESENTING
C EQUAL AREA PORTIONS COVERED BY VCS.  HORIZONTAL SLICES.
C        V of SPHERICAL SECTION     = 1/3*PI*H^2*(3R-H)
C        A of SPHERICAL SECTION     = 2*PI*R*H
C
C NODE   h/R      THETA    BOTTOM/TOP        ALPHA BOTTOM/TOP
C ------------------------------------------------------------
C 1      0.       -.222    0./38.94          1./.9657
C 2      .222     -.444    38.94/56.25       .9657/.8738
C 3      .444     -.667    56.25/70.53       .8738/.7408
C 4      .667     -.889    70.53/83.62       .7408/.5830
C 5      .889     -1.111   83.62/96.38       .5830/.4170
C 6      1.111    -1.333   96.38/109.47      .4170/.2592
C 7      1.333    -1.556   109.47/123.75     .2592/.1262
C 8      1.556    -1.778   123.75/141.06     .1262/.0343
C 9      1.7778   -2.0     141.06/180.       .0343/0.
C
C                          VOL*DEN*CP/NUMNODES
         GEN 1,9,1,43.,.9621*487.*.11/9.     $ FROM BASE UP
C
HEADER CONDUCTOR DATA,TANK
C        K*THK * R * 2PI * SINE(THETA) /LENGTH (= R DELTA THETA)
         1,1,2,8.33*.3/12.*        2.*3.1416*.6285/.4909
         8,8,9,8.33*.3/12.*        2.*3.1416*.6285/.4909
         7,7,8,8.33*.3/12.*        2.*3.1416*.8315/.2757
         2,2,3,8.33*.3/12.*        2.*3.1416*.8315/.2757
         6,6,7,8.33*.3/12.*        2.*3.1416*.9428/.2388
         3,3,4,8.33*.3/12.*        2.*3.1416*.9428/.2388
         5,5,6,8.33*.3/12.*        2.*3.1416*.9938/.2227
         4,4,5,8.33*.3/12.*        2.*3.1416*.9938/.2227
```

```
C VCS: FOLLOW TANK NODAL SCHEME
C
HEADER NODE DATA,VCS
C                       VOL*DEN*CP/NUMNODES
         GEN 1,9,1,350.,.1885*174.*.21/9.     $ FROM BASE UP
C
HEADER CONDUCTOR DATA,VCS
C         K*THK * R * 2PI * SINE(THETA) /LENGTH (= R DELTA THETA)
          1,1,2,133.*.045/12.*       2.*3.1416*.6285/.4909
          8,8,9,133.*.045/12.*       2.*3.1416*.6285/.4909
          7,7,8,133.*.045/12.*       2.*3.1416*.8315/.2757
          2,2,3,133.*.045/12.*       2.*3.1416*.8315/.2757
          6,6,7,133.*.045/12.*       2.*3.1416*.9428/.2388
          3,3,4,133.*.045/12.*       2.*3.1416*.9428/.2388
          5,5,6,133.*.045/12.*       2.*3.1416*.9938/.2227
          4,4,5,133.*.045/12.*       2.*3.1416*.9938/.2227
C CONNECT VCS.1 TO TANK.1 WITH .313" .035 wall 9 to 12"
          10,1,TANK.1,8.*2.7E-4/1.
C
HEADER ARRAY DATA, HYDRO
C
C ARRAY 1: VOID FRACTION AT TOP OF EACH SEGMENT
  1= .9657, .8738, .7408, .5830, .4170, .2592, .1262, .0343, 0.
C ARRAY 2: VOID FRACTION AT BOTTOM OF EACH SEGMENT
  2= 1.0, .9657, .8738, .7408, .5830, .4170, .2592, .1262, .0343
C
C 7000 USER FLUID: PARA HYDROGEN
C
HEADER FLOW DATA,HYDRO,FID=7702
C
C THIS INCLUDES ALL FLUID IN SYSTEM
C
C THIS IS TANK ITSELF: TWO TANKS FOR NONEQ LOGIC.
C INITIAL CONDITIONS: HALF FULL. SATURATED AT 42.6R
C
LU TANK,1,          XL       = 0.0              $ LIQUID HALF
          VOL       = 22.45*0.5
          TL        = 42.6
          PL        = 0.0
LU TANK,2,          XL       = 1.0              $ VAPOR HALF
          VOL       = 22.45*0.5
          TL        = 42.6
          PL        = 1.0E10
PA CONN,1,1,2, GK = 1.0,    DEV      = NULL    $ NONEQ NULL
C
```

```
C DEFAULT PRESSURE IS HEREAFTER 4.0 PSIA
LU DEF, XL        = 1.0
         PL!      = 4.0
         TL       = 42.6
C
PA DEF,  FR       = 0.0382,          WRF      = 1.0E-5
C
C THIS IS ACTUALLY A THROTTLING DEVICE: SET NET PRESSURE DROP USING HC
C
PA CONN,100,1,100,          STAT     = LS,    DEV              = STUBE
         DH       = .18/12.,         TLEN     = 0.5 $ .18"ID
         HC       = -21.   $ WILL RESET: NEED DOWNSTR PRESSURE 4 PSIA
         UPF      = 1.0
LU JUNC,100,                XL       = .1,    TL = 80.0
C
C GENERATE INTERNAL HEAT EXCHANGER
C
M HX,1,D,101,101,101,IHX.1,100,NSEG=6
         TLENT    = 160./12.,        DHS      = 0.18/12.,
         TL       = 28.,             TLINC    = 2.
         XL       = 0.18,            XLINC    = 1.0
         LU       = JUNC,            PA       = STUBEC
C GENERATE TIES TO INTERNAL HEAT EXCHANGER
C HTC=20., INCLUDE FIN EFFECTIVENESSES
C
M GENT,2,HTU,201,1,IHX.1,N=6
         UA       = 20.*0.9*2.22*8./12.
         LUINC    = 0
T HTU,251,1,IHX.201,        UA       = 20.*0.9*1.67
T HTU,252,1,IHX.202,        UA       = 20.*0.9*1.67
T HTU,254,1,IHX.204,        UA       = 20.*0.9*1.67
T HTU,255,1,IHX.205,        UA       = 20.*0.9*1.67
C
C GENERATE TIES TO TANK WALL
C WILL OVERRIDE UA LATER ACCORDING TO FILL LEVEL AND BOILING
C
M GENT,3,HTU,301,1,TANK.1,N=9
         UA       = 20.*38.5/9.
         LUINC    = 0
```

```
C
C NOW GENERATE VCS PIPING
C
PA DEF, DH          = .245/12.
M HX,4,C,401,401,401,VCS.1,106,998,NSEG=9
          TLENT     = 80.,              DHS       = 0.245/12.
          TL        = 325.,             TLINC     = 2.0,     XL       = 1.0
          LU        = JUNC,             PA        = STUBE
C VCS NODE IS ACTUALLY PLATE NOT PIPE, SO INCLUDE
C FIN EFFICIENCY HERE AS AREA FRACTION
          AFRACT    = 0.9
C
C FINALLY, EXHAUST TO "VACUUM"
C THIS IS MAIN CONTROL VALVE TO BALANCE SYSTEM
C
LU JUNC,998,        PL!         = 2.0,    TL        = 400.
PA CONN,998,998,999,            DEV       = MFRSET
PA CONN,999,998,999,            DUP       = 0.0
                                DEV       = LOSS, FK  = 1000.0
C
LU PLEN,999,        TL          = 500.0  $ EXHAUST TO NEAR VACUUM
                    PL!         = 1.0
C         STATE IRRELEVANT SINCE CONNECTED BY MFRSET.
C         THIS PRESSURE IS INTENTIONALLY LOW AND WILL CAUSE ONE WARNING
C         AT THE BEGINNING OF THE PROCESSOR EXECUTION,
C         WHEN THE PRESSURE WILL BE RAISED TO THE MINIMUM ALLOWABLE.
C
C ADD SUBCOOLED FILL LINE AND SOURCE
C
LU PLEN,1999,       TL          = 20.0
                    PL          = 60.0
                    XL          = 0.0
C         THIS TEMP. IS INTENTIONALLY LOW AND WILL CAUSE ONE WARNING
C         AT THE BEGINNING OF THE PROCESSOR EXECUTION, WHEN THE TEMP.
C         WILL BE RAISED TO THE MINIMUM ALLOWABLE.
C
C 10 FOOT BY 0.18" DIA TUBING, INITIALLY MISSING (DUP=0.0 TO TANK #1)
C
PA CONN,1999,1999,1,        DUPJ      = 0.0
          DEV       = STUBE
          DH        = 0.18/12.0
          TLEN      = 10.0
```

```
HEADER OPERATIONS DATA
C
BUILD TVS,TANK,IHX,VCS,MLI
BUILDF TVS, HYDRO
C
C INITIALIZE TANK STATE AND HOLD: APPROX 1/2 FULL
C
DEFMOD HYDRO
         IF(NTEST .NE. 0)GOTO 500
         CALL HLDLMP('HYDRO',1)
         CALL HLDLMP('HYDRO',2)
C
C UPDATE THROTTLING VALVE - REGULATOR
C WANT SPECIFIED DELTAP: USE STUBE HC
C
         HC100              = SNGL(4.0D0 - PL1)
C
C NEGLECT NUCLEATE BOILING RESOLUTION: USE HTRLMP
C TO CONVERT ALL FLOW INTO VAPOR IN FIRST TIE.  THIS
C DUMPS APPROPRIATE AMOUNT OF ENERGY INTO IHX.1
C
         CALL CHGLMP('HYDRO',101,'XL',1.0,'PL')
         CALL HTRLMP('HYDRO',101)
C
C FIND "STEADY STATE" WITH FIXED TANK STATE
C
         CALL FASTIC
         CALL RESAVE('ALL')
500      IF(NTEST .GT. 0) CALL RESTAR(NTEST)
         HTEST    = VHFG(PL1-PATMOS,TL1-ABSZRO,1.0/DL2,HYDRO.FI)
         WRITE(NUSER1,999)FR998*HTEST,QDOT1+QDOT2
F999     FORMAT(' BOIL-OFF ENERGY  = ',1PG13.5,
     .   ', VS. LEAKAGE: ',G13.5/)
         WRITE(NUSER1,1000)
1000     FORMAT(' TIME (MIN)',T16,'PERC. LIQ',T30,'LIQ TEMP',T45
     .   ,'LIQ QUAL',T57,'VAP TEMP',T69,'VAP QUAL',T84,'TVS FR',T95
     .  ,'FILL RATE',T107,'TANK PRES'/)
C
C SWITCH FROM CONSTANT DELTA P THROTTLING TO EQUIVALENT HEAD LOSS
C
         HC100              = 0.0
         FPOW100            = 1.0
         PTEST              = 144.*32.174*3600.**2
         FC100              = PTEST*(PL100-PL1)/FR100**2
         IPDC100            = 0
```

```
C
C SWITCH FROM CONSTANT FLOWRATE EXIT VALVE TO EQUIV HEAD LOSS
C
         DUPI998           = 0.0
         DUPJ998           = 0.0
         DUPI999           = 1.0
         DUPJ999           = 1.0
         FK999    = 2.0*DL998*(AF999/FR998)**2*PTEST*SNGL(PL998-PL999)
         FR999             = FR998
C
C TURN ON FILL LINE (IT APPEARS SUDDENLY TO TANK #1)
C
         DUPJ1999          = 1.0
C
C RELEASE STORAGE TANK FROM BOUNDARY STATE
C
         CALL RELLMP('HYDRO',1)
         CALL RELLMP('HYDRO',2)
C
         TIMEND            = 1.0
         CALL FWDBCK
C
HEADER OUTPUT CALLS, MLI
         IF(NSOL .LE. 1 .AND. LOOPCT .EQ. 0) RETURN
C STEADY STATE AND INFREQUENT TRANSIENT
         CALL LMPTAB('ALL')
         CALL TIETAB('ALL')
         CALL PTHTAB('ALL')
         CALL TPRINT('ALL')
HEADER OUTPUT CALLS, HYDRO
C FREQUENT TRANSIENT (ONE-LINE TO USER FILE)
         IF(NSOL .LE. 1) RETURN
         WRITE(NUSER1,10)TI-
MEN*60.0,VOL1/VTANK,TL1,XL1,TL2,XL2,FR100,FR1999,PL1
10       FORMAT(1X,1P,9G13.5)
C
C IF GREATER THAN 15 MINUTES, PRINT EVERY 3 MINUTES
         IF(TIMEN .GE. 0.2499) OUTPTF        = 3.0/60.0
C IF GREATER THAN 30 MINUTES, PRINT EVERY 5 MINUTES
         IF(TIMEN .GE. 0.4999) OUTPTF        = 5.0/60.0
C
HEADER FLOGIC 0, HYDRO
         IF(LOOPCT .GT. 0.98*NLOOPS) MLI.ITEROT = 1
C
C CALL FIRST HALF OF NONEQ LOGIC
C
         CALL NONEQ0('HYDRO',1,2,1)
```

```
C REPROPORTION TANK WALL TIES ACCORDING TO FILL LEVEL
C USE POOL BOILING IN LIQUID SIDE IF WALL HOTTER
         ATEST              = VOL2/VTANK
C
C LOOP THROUGH TIES TO THE WALL
C
         DO 10 ITEST = 0,8
C ESTIMATE FRACTION OF VAPOR
C ARRAY 1: VOID FRACTION AT TOP OF SEGMENT
C ARRAY 2: VOID FRACTION AT BOTTOM OF SEGMENT
                   JTEST              = ITEST + 1
                   RTEST              = (ATEST-A(1+JTEST))
     .                                  /(A(2+JTEST)-A(1+JTEST))
                   RTEST              = MIN(1.0,MAX(0.0,RTEST))
C CALC SATURATION POINT
                   TSAT      = VTS(PL2-PATMOS,HYDRO.FI) + ABSZRO
C
C IF ALL WITHIN VAPOR TANK: MOVE TIE TO VAPOR, ELSE MOVE TO LIQUID TANK
C
                   IF(RTEST .GE. 1.0)THEN
                         CALL PUTTIE('HYDRO',301+ITEST,2,'TANK',0)
                         IF(TANK.T(1+ITEST) .GE. TSAT)THEN
                              UA(301+ITEST) = AWALL*HTVAP
                         ELSE
                              UA(301+ITEST) = AWALL*HTCON
                         ENDIF
                   ELSE
                         CALL PUTTIE('HYDRO',301+ITEST,1,'TANK',0)
                         IF(TANK.T(1+ITEST) .LE. TSAT)THEN
                              UTEST           = HTLIQ
                         ELSE
                              CALL PBOIL(UTEST,PL2-PATMOS,TSAT-ABSZRO,
     .                          TANK.T(1+ITEST)-ABSZRO,.TRUE.,HYDRO.FI)
                              UTEST           = MAX(HTLIQ,UTEST)
                         ENDIF
C
C IF WITHIN LIQUID SIDE, DAMP HEAT TRANSFER
C
                         UA(301+ITEST) =.5*UA(301+ITEST) + .5*AWALL*(
     .                                   (UTEST*(1.0-RTEST) + RTEST*HTVAP))
                   ENDIF
10       CONTINUE
```

```
HEADER FLOGIC 1, HYDRO
C
C CALL SECOND HALF OF NONEQ LOGIC.
C USE VOL OF TANK (FROM FLOGIC 0) TO CALCULATE SURFACE AREA
C (GROUND TEST: FLAT INTERFACE DUE TO GRAVITY)
C V = PI*H**2/3 * (3R - H), H IS HEIGHT OF SPHERICAL SEGMENT (DEPTH)
C CUBIC FUNCTION OF H:    PI/3 *H**3 - PI*R *H**2 + 0 + V
C USE CUROOT (CHEAT: INTERNAL FLUINT FUNCTION).
C V = PI*H/6 * (3*A**2 + H**2), A IS RADIUS OF ASUR
C RADIUS OF TANK IS RTANK
C
F         HTEST     = CUROOT
     .                        (PI/3.,-PI*RTANK,0.0,VOL2,RTANK,.FALSE.)
          ASUR      = 2.*VOL2/HTEST - PI*HTEST*HTEST/3.
C
C USE UVAP=-2.0 (INCLUDES NATURAL CONVECTION)
C AND ULIQ=-4.0 (INCLUDES STIRRING CAUSED BY INGRESS)
C
          CALL NONEQ1('HYDRO',1,2,1,-4.0,-2.0,ASUR,1.0E30,VTANK)
C
C
HEADER SUBROUTINE DATA
C
FSTART
          SUBROUTINE PBOIL(U,P,T,TW,ENG,IDF)
C
C POOL BOILING CORRELATION FROM ROHSENHOW
C U       RETURNED HEAT TRANS COEF
C P       SAT PRESSURE
C T       SAT TEMP
C TW      WALL TEMP  (TW > T ASSUMED)
C ENG     UNIT FLAG: TRUE IF ENGLISH
C IDF     TRANSLATED PROPERTY ARRAY
C
          DIMENSION IDF(1)
          DOUBLE PRECISION P
          LOGICAL ENG
C
          DATA CSF/0.0132/,GEE/9.806/
          SAVE CSF,GEE
```

```
C CP LIQ
          CPL                 = VCPF(P,T,IDF)
C DEN LIQ
          RHOL                = VDL(T,IDF)
C DEN VAP
          RHOV                = 1.0/VSV(P,T,IDF)
C HEAT OF VAPORIZ.
          HFG                 = VHFG(P,T,1.0/RHOV,IDF)
C VISC LIQ
          VISL                = VVISCF(T,IDF)
C SURF TENS
          STEN                = VST(T,IDF)
C THERMAL COND
          TCONDL              = VCONDF(T,IDF)
C
C CONVERT FOR ENGLISH UNITS
C
          DELT                = TW - T
          IF(ENG)THEN
                    CPL                 = CPL*4187.0
                    RHOL                = RHOL*16.018
                    RHOV                = RHOV*16.018
                    HFG                 = HFG*2326.1
                    VISL                = VISL*1.488/3600.0
                    STEN                = STEN/0.3048*4.448
                    TCONDL              = TCONDL*1.731
                    DELT                = DELT/1.8
          ENDIF
C
C PRANDTL NUMBER TERM
          PRTERM              = CSF*(CPL*VISL/TCONDL)**(1.7)
C
          U                   = ( CPL/PRTERM )**3 *
     .                          ( DELT/HFG )**2 * VISL *
     .                          SQRT( GEE*(RHOL-RHOV)/STEN )
C
C CONVERT BACK TO ENGLISH IF NEEDED
C
          IF(ENG) U           = U/5.678
C
          RETURN
          END
```

```
HEADER FPROP DATA,7702,SI,0.0
C
C MORE COMPLETE PARA H2 TWO-PHASE (FROM 13.8 TO 30K, NEAR 20 K)
C VAPOR PROPERTIES ARE FOR SUPERHEATED VAPOR (1 ATM)
C (EXCEPT CP WHICH MUST BE FOR LOW PRESSURE GAS)
    RGAS  = 8314.34/2.0159
    TCRIT = 32.938,                    PCRIT = 1.2838E6
    ST    = 2.172E-3
    TMIN  = 13.80
    PGMAX = 1.0E6,                     TGMAX = 500.0
C
C *** ALTERNATE INPUTS FOR SATURATED VAPOR PROPERTIES
C IF RUNNING ANALYSES FOR SATURATION CONDITIONS
C THEN USE THE FOLLOWING LINES INSTEAD
C   PGMAX = 1.0E6
C   TGMAX = 32.0
CAT,VG,14.0,0.748E-6,18.0,0.988E-6,  20.27,1.128E-6,  22.0,1.238E-6
C   26.0,1.519E-6,   30.0,1.917E-6,   32.0,2.379E-6
CAT,KG,14.0,0.01254, 18.0,0.01497,   20.27,0.01694,   24.0,0.02181
C   28.0,0.02991,    30.0,0.03787,    32.0,0.06677
C *** END ALTERNATE INPUTS
C
AT,DOME,  13.80,      0.007042E6,      139.75E3 + 309.10E3
C ELSE    24.0,       0.26424E6,       203.16E3 + 213.94E3
          20.28,      0.101325E6,      189.11E3 + 256.33E3
AT,VL,    14.0,24.8E-6,16.0,19.42E-6, 18.0,15.93E-6,   20.0,13.48E-6,
    22.0,11.61E-6,   24.0,10.1E-6,    26.0,8.8E-6,     28.0,7.62E-6,
    30.0,6.46E-6,    32.0,5.13E-6,    32.98,3.54E-6
AT,VG,    14.0,0.748E-6,18.0,0.988E-6,20.27,1.128E-6,  22.0,1.219E-6
    26.0,1.424E-6,   30.0,1.621E-6,   35.0,1.856E-6,   40.0,2.02E-6
    50.0,2.498E-6,   60.0,2.884E-6,   80.0,3.585E-6,   100.0,4.574E-6
    120.0,5.408E-6,  160.0,6.3E-6,    200.0,6.901E-6,  240.0,7.419E-6
    300.0,8.141E-6,  350.0,8.723E-6,  400.0,9.297E-6,  500.0,10.426E-6
AT,KL,    14.0,0.07462,16.0,0.08886,  18.0,0.09543,    20.0,0.0984
    22.0,0.10095,    24.0,0.10084,    26.0,0.09843,    28.0,0.09378
    30.0,0.08657,    32.98,0.09146
AT,KG,    14.0,0.01254,18.0,0.01497,  20.27,0.01694,   24.0,0.0195
    28.0,0.02247,    32.0,0.02534,    35.0,0.02741,    40.0,0.03088
    50.0,0.03781,    60.0,0.04481,    80.0,0.06027,    100.0,0.08954
    140.0,0.13613,   180.0,0.15565,   220.0,0.16341,   260.0,0.16914,
    300.0,0.17591,   400.0,0.19745,   500.0,0.22128
AT,CPG,  14.0,10.54E3,20.0,10.43E3,   24.0,10.39E3,    32.0,10.35E3
    40.0,10.36E3,    50.0,10.49E3,    60.0,10.62E3,    80.0,11.72E3
    100.0,13.4E3,    160.0,16.34E3,   200.0,16.07E3,   280.0,15.0E3
    350.0,14.63E3,   400.0,14.55E3,   500.0,14.52E3
AT,DL,    13.80,77.04,17.0,74.19,     20.28,70.80,     23.0,67.41
    25.0,64.47,      27.0,60.97,      29.0,56.59,      31.0,50.46
    32.0,45.70,      32.938,31.36
```

Processor Output

BOIL-OFF ENERGY = 6.9466 , VS. LEAKAGE: 6.9090

TIME (MIN)	PERC. LIQ	LIQ TEMP	LIQ QUAL	VAP TEMP	VAP QUAL	TVS FR	FILL RATE	TANK PRES
0.	0.50000	42.600	0.	42.600	1.0000	3.82000E-02	217.87	34.642
1.0000	0.53109	41.536	0.	43.639	1.0000	3.99296E-02	203.73	37.636
2.0000	0.56127	40.727	0.	43.984	1.0000	4.13283E-02	191.06	40.147
3.0000	0.58968	40.096	0.	44.375	0.99875	4.27721E-02	176.61	42.823
4.0000	0.61580	39.604	0.	44.957	0.99567	4.43137E-02	159.35	45.776
5.0000	0.63942	39.217	0.	45.485	0.99363	4.57321E-02	141.26	48.579
6.0000	0.66026	38.916	0.	45.952	0.99416	4.69975E-02	122.76	51.152
7.0000	0.67828	38.683	0.	46.352	0.99237	4.80931E-02	104.07	53.444
8.0000	0.69340	38.505	0.	46.680	0.98963	4.90002E-02	85.850	55.373
9.0000	0.70588	38.376	0.	46.932	0.99293	4.97050E-02	68.843	56.894
10.000	0.71572	38.282	0.	47.115	0.99031	5.02188E-02	53.685	58.013
11.000	0.72345	38.219	0.	47.237	0.99041	5.05627E-02	41.064	58.771
12.000	0.72940	38.176	0.	47.313	0.99041	5.07789E-02	31.178	59.248
13.000	0.73401	38.147	0.	47.358	0.98971	5.09074E-02	23.814	59.534
14.000	0.73763	38.128	0.	47.382	0.99016	5.09772E-02	18.947	59.690
15.000	0.74062	38.115	0.	47.397	0.99354	5.10192E-02	15.466	59.783
18.000	0.74477	38.150	0.	47.428	0.99402	5.11067E-02	4.1732	59.978
21.000	0.74678	38.187	0.	47.430	0.99436	5.11131E-02	2.2939	59.992
24.000	0.74824	38.213	0.	47.429	0.99163	5.11130E-02	2.3386	59.992
27.000	0.74972	38.232	0.	47.430	0.99442	5.11130E-02	2.3526	59.992
30.000	0.75115	38.244	0.	47.429	0.99169	5.11124E-02	2.6060	59.990
35.000	0.75371	38.256	0.	47.429	0.99179	5.11117E-02	2.8889	59.988
40.000	0.75639	38.260	0.	47.428	0.99155	5.11108E-02	3.2283	59.986
45.000	0.75917	38.258	0.	47.428	0.99048	5.11110E-02	3.2321	59.986
50.000	0.76203	38.253	0.	47.429	0.98990	5.11111E-02	3.2272	59.986
55.000	0.76497	38.247	0.	47.429	0.99324	5.11113E-02	3.2541	59.985
60.000	0.76785	38.239	0.	47.429	0.99182	5.11115E-02	3.2745	59.985

MODEL = TVS LH2 STORAGE TANK WITH INTERNAL HX AND VAPOR COOLED SHIELD
FASTIC

FLUID SUBMODEL NAME = HYDRO ; FLUID NO. = 7702

LOOPCT = 282
CONVERGENCE STATUS = SUBMODEL CONVERGED AS OF 282 ITERATIONS. ENERGY STABLE BUT UNBALANCED

LUMP PARAMETER TABULATION FOR SUBMODEL HYDRO

LUMP	TYPE	TEMP	PRESSURE	QUALITY	DENSITY	ENTHALPY	HEAT RATE	MASS RATE	ENERGY RATE
1	TANK	42.60	34.64	0.	4.147	-87.32	-2.623	-3.8200E-02	0.7130
2	TANK	42.60	34.64	1.000	0.1773	94.53	9.532	0.	9.532
100	JUNC	29.80	4.000	0.1821	0.1396	-87.32	0.	0.	0.
101	JUNC	29.80	4.000	1.000	2.6064E-02	73.76	6.153	0.	0.
102	JUNC	39.13	4.000	1.000	1.9559E-02	97.58	0.9097	0.	2.1458E-05
103	JUNC	41.70	4.000	1.000	1.8309E-02	104.1	0.2480	0.	-4.9829E-05
104	JUNC	42.37	3.999	1.000	1.8010E-02	105.8	6.4331E-02	0.	-2.6226E-06
105	JUNC	42.54	3.999	1.000	1.7934E-02	106.2	1.6479E-02	0.	3.3855E-05
106	JUNC	42.59	3.999	1.000	1.7914E-02	106.3	4.2725E-03	0.	1.9073E-05
401	JUNC	308.2	3.997	1.000	2.4385E-03	937.0	31.73	0.	-7.6294E-05
402	JUNC	319.0	3.988	1.000	2.3509E-03	978.8	1.547	0.	-5.1843E-02
403	JUNC	321.9	3.979	1.000	2.3246E-03	990.0	0.4053	0.	-2.0961E-02
404	JUNC	323.7	3.970	1.000	2.3062E-03	997.0	0.2093	0.	-5.9659E-02
405	JUNC	325.1	3.961	1.000	2.2914E-03	1002.	0.1693	0.	-2.8759E-02
406	JUNC	326.1	3.951	1.000	2.2785E-03	1006.	9.8081E-02	0.	-5.9463E-02
407	JUNC	327.0	3.942	1.000	2.2669E-03	1010.	0.1045	0.	-2.8656E-02
408	JUNC	327.8	3.933	1.000	2.2561E-03	1013.	6.0936E-02	0.	-5.4210E-02
409	JUNC	328.7	3.923	1.000	2.2444E-03	1016.	0.1128	0.	-2.1418E-02
998	JUNC	328.7	3.918	1.000	2.2416E-03	1016.	0.	0.	-1.8883E-03
999	PLEN	500.0	1.022	1.000	3.8440E-04	1657.	0.	3.8200E-02	38.83
1999	PLEN	24.84	60.00	0.	4.809	-128.0	0.	-217.9	2.7886E+04

MODEL = TVS
FASTIC

LH2 STORAGE TANK WITH INTERNAL HX AND VAPOR COOLED SHIELD

FLUID SUBMODEL NAME = HYDRO ; FLUID NO. = 7702

LOOPCT = 282
CONVERGENCE STATUS = SUBMODEL CONVERGED AS OF 282 ITERATIONS. ENERGY STABLE BUT UNBALANCED

TIE PARAMETER TABULATION FOR SUBMODEL HYDRO

TIE	TYPE	UA	QTIE	LUMP	TEMP.	NODE	TEMP.	PATH 1	FRACT	PATH 2	FRACT
101	HTN	0.2080	6.153	101	29.80	IHX.1	42.38	101	1.0000		
102	HTN	0.2649	0.9097	102	39.13	IHX.2	42.56	102	1.0000		
103	HTN	0.2794	0.2480	103	41.70	IHX.3	42.59	103	1.0000		
104	HTN	0.2832	6.4331E-02	104	42.37	IHX.4	42.60	104	1.0000		
105	HTN	0.2841	1.6479E-02	105	42.54	IHX.5	42.60	105	1.0000		
106	HTN	0.2844	4.2725E-03	106	42.59	IHX.6	42.60	106	1.0000		
201	HTU	26.64	-5.904	1	42.60	IHX.1	42.38				
202	HTU	26.64	-0.9538	1	42.60	IHX.2	42.56				
203	HTU	26.64	-0.2562	1	42.60	IHX.3	42.59				
204	HTU	26.64	-6.4888E-02	1	42.60	IHX.4	42.60				
205	HTU	26.64	-1.6388E-02	1	42.60	IHX.5	42.60				
206	HTU	26.64	9.5901E-03	1	42.60	IHX.6	42.60				
251	HTU	30.06	-0.1493	1	42.60	IHX.201	42.60				
252	HTU	30.06	-2.5753E-02	1	42.60	IHX.202	42.60				
254	HTU	30.06	-1.6403E-03	1	42.60	IHX.204	42.60				
255	HTU	30.06	-4.3106E-04	1	42.60	IHX.205	42.60				
301	HTU	85.56	1.267	1	42.60	TANK.1	42.61				
302	HTU	85.56	0.8048	1	42.60	TANK.2	42.61				
303	HTU	85.56	0.8251	1	42.60	TANK.3	42.61				
304	HTU	85.56	0.8857	1	42.60	TANK.4	42.61				
305	HTU	51.33	0.9576	1	42.60	TANK.5	42.62				
306	HTU	17.11	0.7621	2	42.60	TANK.6	42.64				
307	HTU	17.11	0.9182	2	42.60	TANK.7	42.65				
308	HTU	17.11	1.307	2	42.60	TANK.8	42.68				
309	HTU	17.11	6.544	2	42.60	TANK.9	42.98				
401	HTNC	8.044	31.73	401	308.2	VCS.1	312.2	401	0.9000	402	0.4500
402	HTNC	8.200	1.547	402	319.0	VCS.2	319.2	402	0.4500	403	0.4500
403	HTNC	8.241	0.4053	403	321.9	VCS.3	322.0	403	0.4500	404	0.4500
404	HTNC	8.267	0.2093	404	323.7	VCS.4	323.8	404	0.4500	405	0.4500
405	HTNC	8.277	0.1693	405	325.1	VCS.5	325.1	405	0.4500	406	0.4500
406	HTNC	8.284	9.8081E-02	406	326.1	VCS.6	326.1	406	0.4500	407	0.4500
407	HTNC	8.289	0.1045	407	327.0	VCS.7	327.0	407	0.4500	408	0.4500
408	HTNC	8.293	6.0936E-02	408	327.8	VCS.8	327.8	408	0.4500	409	0.4500
409	HTNC	8.298	0.1128	409	328.7	VCS.9	328.7	409	0.4500	410	0.9000

MODEL = TVS
FASTIC

LH2 STORAGE TANK WITH INTERNAL HX AND VAPOR COOLED SHIELD

FLUID SUBMODEL NAME = HYDRO ; FLUID NO. = 7702

LOOPCT = 282
CONVERGENCE STATUS = SUBMODEL CONVERGED AS OF 282 ITERATIONS. ENERGY STABLE BUT UNBALANCED

PATH PARAMETER TABULATION FOR SUBMODEL HYDRO

PATH	TYPE	LMP 1	LMP 2	DUP I	DUP J	STAT	XL UPSTRM	FLOWRATE	DELTA PRES	FLOW REGIMES OR REYNOLDS NO.	
1	NULL	1	2	1.0	1.0	NORM	0.	0.	0.		
100	STUBE	1	100	1.0	1.0	LS	0.	3.8200E-02	30.64	129.47	
101	STUBE	100	101	1.0	1.0	NORM	0.1821	3.8200E-02	1.1191E-04	UNKNOWN	1486.7
102	STUBE	101	102	1.0	1.0	NORM	1.000	3.8200E-02	1.4235E-04	1486.7	1112.0
103	STUBE	102	103	1.0	1.0	NORM	1.000	3.8200E-02	1.8367E-04	1112.0	1048.0
104	STUBE	103	104	1.0	1.0	NORM	1.000	3.8200E-02	1.9604E-04	1048.0	1032.6
105	STUBE	104	105	1.0	1.0	NORM	1.000	3.8200E-02	1.9936E-04	1032.6	1028.8
106	STUBE	105	106	1.0	1.0	NORM	1.000	3.8200E-02	2.0017E-04	1028.8	1027.8
401	STUBE	106	401	1.0	1.0	NORM	1.000	3.8200E-02	2.2666E-03	755.11	152.21
402	STUBE	401	402	1.0	1.0	NORM	1.000	3.8200E-02	8.7353E-03	152.21	150.12
403	STUBE	402	403	1.0	1.0	NORM	1.000	3.8200E-02	9.0202E-03	150.12	149.58
404	STUBE	403	404	1.0	1.0	NORM	1.000	3.8200E-02	9.1338E-03	149.58	149.23
405	STUBE	404	405	1.0	1.0	NORM	1.000	3.8200E-02	9.2160E-03	149.23	148.98
406	STUBE	405	406	1.0	1.0	NORM	1.000	3.8200E-02	9.2878E-03	148.98	148.78
407	STUBE	406	407	1.0	1.0	NORM	1.000	3.8200E-02	9.3494E-03	148.78	148.61
408	STUBE	407	408	1.0	1.0	NORM	1.000	3.8200E-02	9.4057E-03	148.61	148.46
409	STUBE	408	409	1.0	1.0	NORM	1.000	3.8200E-02	9.4626E-03	148.46	148.30
410	STUBE	409	998	1.0	1.0	NORM	1.000	3.8200E-02	4.7489E-03	148.30	148.29
998	MFRSET	998	999	1.0	1.0	NORM	1.000	3.8200E-02	2.896		
999	LOSS	998	999	0.	0.	NORM	1.000	0.2891	2.896		
1999	STUBE	1999	1	1.0	0.	NORM	0.	217.9	25.36	3.01685E+05	7.38441E+05

MODEL = TVS
FASTIC

LH2 STORAGE TANK WITH INTERNAL HX AND VAPOR COOLED SHIELD

SUBMODEL NAME = IHX

		CALCULATED			ALLOWED	
MAX DIFF DELTA T PER ITER	DRLXCC(	0)=	0.	VS.	DRLXCA=	1.000000E-02
MAX ARITH DELTA T PER ITER	ARLXCC(IHX	205)=	-3.814697E-06	VS.	ARLXCA=	1.000000E-02
MAX SYSTEM ENERGY BALANCE	EBALSC	=	4.006898E-04	VS.	EBALSA * ESUMIS =	7.372428E-02
					EBALSA=	1.000000E-02
ENERGY INTO AND OUT OF SYS	ESUMIS	=	7.37243		ESUMOS=	7.40562
MAX NODAL ENERGY BALANCE	EBALNC(IHX	4)=	1.614839E-04	VS.	EBALNA=	0.
NUMBER OF ITERATIONS	LOOPCT	=	282	VS.	NLOOPS=	500
PROBLEM TIME	TIMEN	=	0.	VS.	TIMEND=	0.

DIFFUSION NODES IN INPUT NODE NUMBER ORDER
←NONE→
ARITHMETIC NODES IN INPUT NODE NUMBER ORDER

T 1=	42.378	T 2=	42.564	T 3=	42.590	T 4=	42.598	T 5=	42.599	T 6=	42.600		
T 201=	42.595	T 202=	42.599	T 204=	42.600	T 205=	42.600						

HEATER NODES IN INPUT NODE NUMBER ORDER
←NONE→
BOUNDARY NODES IN INPUT NODE NUMBER ORDER
←NONE→

MODEL = TVS
FASTIC

LH2 STORAGE TANK WITH INTERNAL HX AND VAPOR COOLED SHIELD

SUBMODEL NAME = TANK

		CALCULATED			ALLOWED	
MAX DIFF DELTA T PER ITER	DRLXCC(TANK	9)=	-1.144409E-05	VS.	DRLXCA=	1.000000E-02
MAX ARITH DELTA T PER ITER	ARLXCC(	0)=	0.	VS.	ARLXCA=	1.000000E-02
MAX SYSTEM ENERGY BALANCE	EBALSC	=	-3.437400E-04	VS.	EBALSA * ESUMIS =	0.
					EBALSA=	1.000000E-02
ENERGY INTO AND OUT OF SYS	ESUMIS	=	0.		ESUMOS=	14.2695
MAX NODAL ENERGY BALANCE	EBALNC(TANK	7)=	-2.157092E-04	VS.	EBALNA=	0.
NUMBER OF ITERATIONS	LOOPCT	=	282	VS.	NLOOPS=	500
PROBLEM TIME	TIMEN	=	0.	VS.	TIMEND=	0.

DIFFUSION NODES IN INPUT NODE NUMBER ORDER

T 1=	42.615	T 2=	42.609	T 3=	42.610	T 4=	42.610	T 5=	42.619	T 6=	42.645		
T 7=	42.654	T 8=	42.676	T 9=	42.982								

ARITHMETIC NODES IN INPUT NODE NUMBER ORDER
←NONE→
HEATER NODES IN INPUT NODE NUMBER ORDER
←NONE→
BOUNDARY NODES IN INPUT NODE NUMBER ORDER
←NONE→

MODEL = TVS
FASTIC

LB2 STORAGE TANK WITH INTERNAL HX AND VAPOR COOLED SHIELD

SUBMODEL NAME = VCS

	CALCULATED			ALLOWED	
MAX DIFF DELTA T PER ITER	DRLXCC(VCS	9)=-7.904053E-03	VS.	DRLXCA=	1.000000E-02
MAX ARITH DELTA T PER ITER	ARLXCC(	0)= 0.	VS.	ARLXCA=	1.000000E-02
MAX SYSTEM ENERGY BALANCE	EBALSC	=-0.535101	VS.	EBALSA = ESUMIS	= 0.
				EBALSA=	1.000000E-02
ENERGY INTO AND OUT OF SYS	ESUMIS	= 0.		ESUMOS=	34.4656
MAX NODAL ENERGY BALANCE	EBALNC(VCS	5)=-9.419680E-02	VS.	EBALNA=	0.
NUMBER OF ITERATIONS	LOOPCT	= 282	VS.	NLOOPS=	500
PROBLEM TIME	TIMEN	= 0.	VS.	TIMEND=	0.

DIFFUSION NODES IN INPUT NODE NUMBER ORDER

T 1= 312.18 T 2= 319.22 T 3= 321.95 T 4= 323.75 T 5= 325.08 T 6= 326.15
T 7= 327.04 T 8= 327.82 T 9= 328.73

ARITHMETIC NODES IN INPUT NODE NUMBER ORDER

++NONE++

HEATER NODES IN INPUT NODE NUMBER ORDER

++NONE++

BOUNDARY NODES IN INPUT NODE NUMBER ORDER

++NONE++

MODEL = TVS
FASTIC

LB2 STORAGE TANK WITH INTERNAL HX AND VAPOR COOLED SHIELD

SUBMODEL NAME = MLI

	CALCULATED			ALLOWED	
MAX DIFF DELTA T PER ITER	DRLXCC(	0)= 0.	VS.	DRLXCA=	1.000000E-02
MAX ARITH DELTA T PER ITER	ARLXCC(MLI	9)=-3.051758E-04	VS.	ARLXCA=	1.000000E-02
MAX SYSTEM ENERGY BALANCE	EBALSC	= 4.207412E-05	VS.	EBALSA = ESUMIS	= 0.482335
				EBALSA=	1.000000E-02
ENERGY INTO AND OUT OF SYS	ESUMIS	= 48.2335		ESUMOS=	0.
MAX NODAL ENERGY BALANCE	EBALNC(MLI	7)= 1.402470E-05	VS.	EBALNA=	0.
NUMBER OF ITERATIONS	LOOPCT	= 282	VS.	NLOOPS=	500
PROBLEM TIME	TIMEN	= 0.	VS.	TIMEND=	0.

DIFFUSION NODES IN INPUT NODE NUMBER ORDER

++NONE++

ARITHMETIC NODES IN INPUT NODE NUMBER ORDER

T 1= 532.85 T 2= 532.93 T 3= 532.97 T 4= 532.99 T 5= 533.01 T 6= 533.02
T 7= 533.04 T 8= 533.05 T 9= 533.06

HEATER NODES IN INPUT NODE NUMBER ORDER

++NONE++

BOUNDARY NODES IN INPUT NODE NUMBER ORDER

T 999= 540.00

RESTART RECORD NUMBER 203 WRITTEN FOR TIMEN = 0.

SUBMODEL NAME = IHX

ARLXCA =	1.000000E-02	ARLXCC =	0.	ATMPCA =	1.000000E+30	ATMPCC =	0.	BACKUP =	0.
CSGFAC =	0.	CSGMAX =	0.	CSGMIN =	0.	DRLXCA =	1.000000E-02	DRLXCC =	0.
DTIMEH =	1.000000E+30	DTIMEI =	0.	DTIMEL =	0.	DTIMEU =	0.	DTMPCA =	1.000000E+30
DTMPCC =	0.	EBALNA =	0.	EBALNC =	1.614839E-04	EBALSA =	1.000000E-02	EBALSC =	0.
ESUMIS =	7.37243	ESUMOS =	7.40562	EXTLIM =	5.00000	ITHOLD =	10	NARLXN =	0
NATMPN =	0	NCGMAN =	0	NCSGMN =	0	NDRLXN =	0	NDTMPN =	0
NEBALN =	0	NLOOPT =	100	ITEROT =	0	ITERXT =	6	OPEITR =	0
OUTPUT =	0.100000								

MODEL = TVS
FWDBCK

LH2 STORAGE TANK WITH INTERNAL HX AND VAPOR COOLED SHIELD

FLUID SUBMODEL NAME = HYDRO ; FLUID NO. = 7702

MAX TIME STEP (GROWTH LIMITED) = 0.191447 ; LAST CAUSE: TANK 1; TEMP./DENSITY CHANGE LIMIT (WAS 2.3931E-02)
LAST TIME STEP = 2.719462E-07 VS. DTMAXF/DTMINF = 1.000000E+30 / 0. ; AVERAGE TIME STEP = 3.344411E-03
PROBLEM TIME TIMEN = 1.00000 VS. TIMEND = 1.00000

LUMP PARAMETER TABULATION FOR SUBMODEL HYDRO

LUMP	TYPE	TEMP	PRESSURE	QUALITY	DENSITY	ENTHALPY	HEAT RATE	MASS RATE	ENERGY RATE
1	TANK	38.24	59.99	0.	4.345	-98.22	54.28	3.322	-356.4
2	TANK	47.43	59.99	0.9918	0.2988	96.65	-23.30	-9.8185E-02	-24.46
100	JUNC	30.90	5.129	0.1143	0.2692	-98.22	0.	0.	0.
101	JUNC	30.90	5.128	0.9885	3.2809E-02	73.76	8.790	0.	9.5367E-07
102	JUNC	35.32	5.128	1.000	2.8089E-02	87.43	0.6982	0.	-2.3842E-07
103	JUNC	37.24	5.128	1.000	2.6554E-02	92.34	0.2510	3.7253E-09	4.7684E-07
104	JUNC	37.90	5.128	1.000	2.6065E-02	94.02	8.5938E-02	-3.7253E-09	0.
105	JUNC	38.12	5.128	1.000	2.5903E-02	94.59	2.9053E-02	0.	0.
106	JUNC	38.20	5.128	1.000	2.5849E-02	94.78	9.7656E-03	0.	0.
401	JUNC	302.0	5.125	1.000	3.1911E-03	912.8	41.81	0.	0.
402	JUNC	316.0	5.116	1.000	3.0450E-03	967.0	2.769	0.	0.
403	JUNC	319.8	5.107	1.000	3.0035E-03	981.6	0.7486	0.	0.
404	JUNC	322.1	5.098	1.000	2.9763E-03	990.7	0.4612	0.	0.
405	JUNC	323.8	5.088	1.000	2.9552E-03	997.2	0.3341	0.	0.
406	JUNC	325.1	5.078	1.000	2.9376E-03	1002.	0.2614	0.	0.
407	JUNC	326.2	5.069	1.000	2.9221E-03	1007.	0.2192	0.	0.
408	JUNC	327.2	5.059	1.000	2.9079E-03	1010.	0.1907	0.	0.
409	JUNC	328.3	5.049	1.000	2.8926E-03	1015.	0.2159	0.	0.
998	JUNC	328.3	5.044	1.000	2.8897E-03	1015.	0.	0.	0.
999	PLEN	500.0	1.022	1.000	3.8440E-04	1657.	0.	5.1112E-02	51.86
1999	PLEN	24.84	60.00	0.	4.809	-128.0	0.	-3.274	419.1

MODEL = TVS
FWDBCK

LH2 STORAGE TANK WITH INTERNAL HX AND VAPOR COOLED SHIELD

FLUID SUBMODEL NAME = HYDRO ; FLUID NO. = 7702

MAX TIME STEP (GROWTH LIMITED) = 0.191447 ; LAST CAUSE: TANK 1; TEMP./DENSITY CHANGE LIMIT (WAS 2.3931E-02)
LAST TIME STEP = 2.719462E-07 VS. DTMAXF/DTMINF = 1.000000E+30 / 0. ; AVERAGE TIME STEP = 3.344411E-03
PROBLEM TIME TIMEN = 1.00000 VS. TIMEND = 1.00000

TIE PARAMETER TABULATION FOR SUBMODEL HYDRO

TIE	TYPE	UA	QTIE	LUMP	TEMP.	NODE	TEMP.	PATH 1	FRACT	PATH 2	FRACT
101	HTN	13.22	8.790	101	30.90	HX.1	37.92	101	1.0000		
102	HTN	0.2418	0.6982	102	35.32	HX.2	38.21	102	1.0000		
103	HTN	0.2543	0.2510	103	37.24	HX.3	38.23	103	1.0000		
104	HTN	0.2580	8.5938E-02	104	37.90	HX.4	38.24	104	1.0000		
105	HTN	0.2593	2.9053E-02	105	38.12	HX.5	38.24	105	1.0000		
106	HTN	0.2597	9.7656E-03	106	38.20	HX.6	38.24	106	1.0000		
201	HTU	26.64	-8.425	1	38.24	HX.1	37.92				
202	HTU	26.64	-0.7923	1	38.24	HX.2	38.21				
203	HTU	26.64	-0.2568	1	38.24	HX.3	38.23				
204	HTU	26.64	-8.5575E-02	1	38.24	HX.4	38.24				
205	HTU	26.64	-2.8629E-02	1	38.24	HX.5	38.24				
206	HTU	26.64	9.3460E-03	1	38.24	HX.6	38.24				
251	HTU	30.06	-0.2130	1	38.24	HX.201	38.23				
252	HTU	30.06	-2.2381E-02	1	38.24	HX.202	38.24				
254	HTU	30.06	-2.1667E-03	1	38.24	HX.204	38.24				
255	HTU	30.06	-7.4387E-04	1	38.24	HX.205	38.24				
301	HTU	85.56	1.744	1	38.24	TANK.1	38.26				
302	HTU	85.56	1.308	1	38.24	TANK.2	38.25				
303	HTU	85.56	1.337	1	38.24	TANK.3	38.25				
304	HTU	85.56	1.380	1	38.24	TANK.4	38.26				
305	HTU	85.56	1.712	1	38.24	TANK.5	38.26				
306	HTU	85.56	6.555	1	38.24	TANK.6	38.32				
307	HTU	31.03	33.21	1	38.24	TANK.7	39.31				
308	HTU	85.56	-28.49	2	47.43	TANK.8	47.10				
309	HTU	17.11	5.195	2	47.43	TANK.9	47.73				
401	HTNC	7.955	41.81	401	302.0	VCS.1	307.3	401	0.9000	402	0.4500
402	HTNC	8.156	2.769	402	316.0	VCS.2	316.3	402	0.4500	403	0.4500
403	HTNC	8.210	0.7486	403	319.8	VCS.3	319.8	403	0.4500	404	0.4500
404	HTNC	8.244	0.4612	404	322.1	VCS.4	322.1	404	0.4500	405	0.4500
405	HTNC	8.268	0.3341	405	323.8	VCS.5	323.8	405	0.4500	406	0.4500
406	HTNC	8.278	0.2614	406	325.1	VCS.6	325.1	406	0.4500	407	0.4500
407	HTNC	8.284	0.2192	407	326.2	VCS.7	326.2	407	0.4500	408	0.4500
408	HTNC	8.289	0.1907	408	327.2	VCS.8	327.2	408	0.4500	409	0.4500
409	HTNC	8.296	0.2159	409	328.3	VCS.9	328.3	409	0.4500	410	0.9000

MODEL = TVS
FWDBCK

LB2 STORAGE TANK WITH INTERNAL HX AND VAPOR COOLED SHIELD

FLUID SUBMODEL NAME = HYDRO ; FLUID NO. = 7702

MAX TIME STEP (GROWTH LIMITED) = 0.191447 ; LAST CAUSE: TANK 1; TEMP./DENSITY CHANGE LIMIT (WAS 2.3931E-02)
LAST TIME STEP = 2.719462E-07 VS. DTMAXF/DTMINF = 1.000000E+30 / 0. ; AVERAGE TIME STEP = 3.344411E-03
PROBLEM TIME TIMEN = 1.00000 VS. TIMEND = 1.00000

PATH PARAMETER TABULATION FOR SUBMODEL HYDRO

PATH	TYPE	LMP 1	LMP 2	DUP I	DUP J	STAT	XL UPSTRM	FLOWRATE	DELTA PRES	FLOW REGIMES OR REYNOLDS NO.	
1	NULL	1	2	1.0	1.0	NORM	0.9918	-9.8185E-02	-1.4469E-03		
100	STUBE	1	100	1.0	1.0	LS	0.	5.1112E-02	54.86	145.59	
101	STUBE	100	101	1.0	1.0	NORM	0.1143	5.1112E-02	1.3012E-04	UNKNOWN	UNKNOWN
102	STUBE	101	102	1.0	1.0	NORM	0.9885	5.1112E-02	1.3171E-04	UNKNOWN	1648.2
103	STUBE	102	103	1.0	1.0	NORM	1.000	5.1112E-02	1.5398E-04	1648.2	1559.3
104	STUBE	103	104	1.0	1.0	NORM	1.000	5.1112E-02	1.6338E-04	1559.3	1533.6
105	STUBE	104	105	1.0	1.0	NORM	1.000	5.1112E-02	1.6658E-04	1533.6	1525.1
106	STUBE	105	106	1.0	1.0	NORM	1.000	5.1112E-02	1.6756E-04	1525.1	1522.3
401	STUBE	106	401	1.0	1.0	NORM	1.000	5.1112E-02	2.3184E-03	1118.4	205.31
402	STUBE	401	402	1.0	1.0	NORM	1.000	5.1112E-02	8.9275E-03	205.31	201.65
403	STUBE	402	403	1.0	1.0	NORM	1.000	5.1112E-02	9.2986E-03	201.65	200.68
404	STUBE	403	404	1.0	1.0	NORM	1.000	5.1112E-02	9.4408E-03	200.68	200.09
405	STUBE	404	405	1.0	1.0	NORM	1.000	5.1112E-02	9.5416E-03	200.09	199.66
406	STUBE	405	406	1.0	1.0	NORM	1.000	5.1112E-02	9.6224E-03	199.66	199.33
407	STUBE	406	407	1.0	1.0	NORM	1.000	5.1112E-02	9.6917E-03	199.33	199.05
408	STUBE	407	408	1.0	1.0	NORM	1.000	5.1112E-02	9.7538E-03	199.05	198.81
409	STUBE	408	409	1.0	1.0	NORM	1.000	5.1112E-02	9.8160E-03	198.81	198.53
410	STUBE	409	998	1.0	1.0	NORM	1.000	5.1112E-02	4.9263E-03	198.53	198.53
998	MFRSET	998	999	0.	0.	NORM	1.000	3.8200E-02	4.022		
999	LOSS	998	999	1.0	1.0	NORM	1.000	5.1112E-02	4.022		
1999	STUBE	1999	1	1.0	1.0	NORM	0.	3.274	1.4675E-02	4534.1	9327.3

MODEL = TVS
FWDBCK

LB2 STORAGE TANK WITH INTERNAL HX AND VAPOR COOLED SHIELD

SUBMODEL NAME = IHX

	CALCULATED		ALLOWED
MAX DIFF DELTA T PER ITER	DRLXCC(0)=	0.	VS. DRLXCA= 1.000000E-02
MAX ARITH DELTA T PER ITER	ARLXCC(IHX 5)=	-7.629395E-06	VS. ARLXCA= 1.000000E-02
MAX DIFF DEL T PER TIME STEP	DTMPCC(0)=	0.	VS. DTMPCA= 1.000000E+30
MAX ARITH DEL T PER TIME STEP	ATMPCC(IHX 5)=	-7.629395E-06	VS. ATMPCA= 1.000000E+30
MIN STABILITY CRITERIA	CSGMIN(0)=	1.000000E+30	
MAX STABILITY CRITERIA	CSGMAX(0)=	-1.000000E+30	
NUMBER OF ITERATIONS	LOOPCT =	1	VS. NLOOPT= 100
PROBLEM TIME	TIMEN =	1.00000	VS. TIMEND= 1.00000
MEAN PROBLEM TIME	TIMEM =	1.00000	
AVERAGE TIME STEP USED SINCE LAST OUTPUT	=	4.999998E-03	VS. DTIMEI= 0.

DIFFUSION NODES IN INPUT NODE NUMBER ORDER
↔NONE↔

ARITHMETIC NODES IN INPUT NODE NUMBER ORDER

T	1= 37.923	T	2= 38.209	T	3= 38.229	T	4= 38.236	T	5= 38.238	T	6= 38.239
T	201= 38.232	T	202= 38.238	T	204= 38.239	T	205= 38.239				

HEATER NODES IN INPUT NODE NUMBER ORDER
↔NONE↔

BOUNDARY NODES IN INPUT NODE NUMBER ORDER
↔NONE↔

MODEL = TVS
FWDBCK

LH2 STORAGE TANK WITH INTERNAL HX AND VAPOR COOLED SHIELD

SUBMODEL NAME = TANK

	CALCULATED			ALLOWED	
MAX DIFF DELTA T PER ITER	DRLXCC(TANK	9)=	7.629395E-06 VS.	DRLXCA=	1.000000E-02
MAX ARITH DELTA T PER ITER	ARLXCC(	0)=	0. VS.	ARLXCA=	1.000000E-02
MAX DIFF DEL T PER TIME STEP	DTMPCC(TANK	9)=	7.629395E-06 VS.	DTMPCA=	1.000000E+30
MAX ARITH DEL T PER TIME STEP	ATMPCC(	0)=	0. VS.	ATMPCA=	1.000000E+30
MIN STABILITY CRITERIA	CSGMIN(TANK	5)=	5.889374E-02		
MAX STABILITY CRITERIA	CSGMAX(TANK	9)=	0.304577		
NUMBER OF ITERATIONS	LOOPCT	=	1 VS.	NLOOPT=	100
PROBLEM TIME	TIMEN	=	1.00000 VS.	TIMEND=	1.00000
MEAN PROBLEM TIME	TIMEM	=	1.00000		
AVERAGE TIME STEP USED SINCE LAST OUTPUT		=	4.999998E-03 VS.	DTIMEI=	0.

DIFFUSION NODES IN INPUT NODE NUMBER ORDER

T	1=	38.259	T	2=	38.254	T	3=	38.255	T	4=	38.255	T	5=	38.259	T	6=	38.316
T	7=	39.309	T	8=	47.096	T	9=	47.733									

ARITHMETIC NODES IN INPUT NODE NUMBER ORDER
↔NONE↔

HEATER NODES IN INPUT NODE NUMBER ORDER
↔NONE↔

BOUNDARY NODES IN INPUT NODE NUMBER ORDER
↔NONE↔

MODEL = TVS
FWDBCK

LH2 STORAGE TANK WITH INTERNAL HX AND VAPOR COOLED SHIELD

SUBMODEL NAME = VCS

	CALCULATED			ALLOWED	
MAX DIFF DELTA T PER ITER	DRLXCC(VCS	9)=	-3.051758E-05 VS.	DRLXCA=	1.000000E-02
MAX ARITH DELTA T PER ITER	ARLXCC(	0)=	0. VS.	ARLXCA=	1.000000E-02
MAX DIFF DEL T PER TIME STEP	DTMPCC(VCS	9)=	-3.051758E-05 VS.	DTMPCA=	1.000000E+30
MAX ARITH DEL T PER TIME STEP	ATMPCC(	0)=	0. VS.	ATMPCA=	1.000000E+30
MIN STABILITY CRITERIA	CSGMIN(VCS	5)=	2.110499E-02		
MAX STABILITY CRITERIA	CSGMAX(VCS	1)=	6.381326E-02		
NUMBER OF ITERATIONS	LOOPCT	=	1 VS.	NLOOPT=	100
PROBLEM TIME	TIMEN	=	1.00000 VS.	TIMEND=	1.00000
MEAN PROBLEM TIME	TIMEM	=	1.00000		
AVERAGE TIME STEP USED SINCE LAST OUTPUT		=	4.999998E-03 VS.	DTIMEI=	0.

DIFFUSION NODES IN INPUT NODE NUMBER ORDER

T	1=	307.29	T	2=	316.33	T	3=	319.65	T	4=	322.14	T	5=	323.81	T	6=	325.14
T	7=	326.24	T	8=	327.19	T	9=	328.28									

ARITHMETIC NODES IN INPUT NODE NUMBER ORDER
↔NONE↔

HEATER NODES IN INPUT NODE NUMBER ORDER
↔NONE↔

BOUNDARY NODES IN INPUT NODE NUMBER ORDER
↔NONE↔

MODEL = TVS
FWDBCK

LB2 STORAGE TANK WITH INTERNAL BX AND VAPOR COOLED SHIELD

SUBMODEL NAME = MLI

	CALCULATED			ALLOWED	
MAX DIFF DELTA T PER ITER	DRLXCC(	0)=	0.	VS. DRLXCA=	1.000000E-02
MAX ARITH DELTA T PER ITER	ARLXCC(MLI	9)=	0.	VS. ARLXCA=	1.000000E-02
MAX DIFF DEL T PER TIME STEP	DTMPCC(	0)=	0.	VS. DTMPCA=	1.000000E+30
MAX ARITH DEL T PER TIME STEP	ATMPCC(MLI	9)=	0.	VS. ATMPCA=	1.000000E+30
MIN STABILITY CRITERIA	CSGMIN(	0)=	1.000000E+30		
MAX STABILITY CRITERIA	CSGMAX(	0)=	-1.000000E+30		
NUMBER OF ITERATIONS	LOOPCT	=	1	VS. NLOOPT=	100
PROBLEM TIME	TIMEN	=	1.00000	VS. TIMEND=	1.00000
MEAN PROBLEM TIME	TIMEM	=	1.00000		
AVERAGE TIME STEP USED SINCE LAST OUTPUT		=	4.999998E-03	VS. DTIMEI=	0.

DIFFUSION NODES IN INPUT NODE NUMBER ORDER
++NONE++

ARITHMETIC NODES IN INPUT NODE NUMBER ORDER

T	1= 532.79	T	2= 532.90	T	3= 532.94	T	4= 532.97	T	5= 532.99	T	6= 533.01
T	7= 533.02	T	8= 533.04	T	9= 533.05						

HEATER NODES IN INPUT NODE NUMBER ORDER
++NONE++

BOUNDARY NODES IN INPUT NODE NUMBER ORDER

T 999= 540.00

PROBLEM G: HEAT PUMP RADIATOR

With the exception of open cycle systems, all spacecraft must ultimately cope with radiation to space as the sole means of rejecting waste heat. The size of the surface area required to reject a given amount of heat is a function of the operating temperature, emissivity at that temperature (usually infrared), solar absorptivity, and space environments (e.g., views to other surfaces, planets, suns, etc.). Of these, the dominant term is usually operating temperature because of the fourth power dependence, but this is also the parameter that can be varied the least: its value is usually tightly constrained by the electronics, humans, or other payloads that are rejecting heat.

A long standing temptation is to boost the rejection temperature using a heat pump cycle, reducing radiator area at the expense of added power (e.g., electric power to a compressor), added complexity, and reduced reliability. Thus far, that trade has been unacceptable except in cases where there is no choice: the payload requires a temperature that is less than that of the average rejection environment.

When the rejected power cannot be radiated through existing spacecraft surfaces, a remote radiator is required. Frequently, energy is transported to this radiator by heat pipes or fluid loops, and/or the radiator is isothermalized with heat pipes or fluid loops. If a vapor compression heat pump is used to overcome an adverse temperature gradient, it can fulfill the first requirement (transport), but should it also be used to fulfill the second requirement (radiator isothermalization by direct condensation)? While a direct condensation radiator features improved efficiency, lower temperature drops, and less weight compared to the more traditional approach of heat exchanging to a heat pipe manifold, it can complicate the plumbing design and, more important, it exposes the working fluid to micrometeoroid/debris environments and potentially to freezing temperatures, requiring defensive design measures such as bumpers, auxiliary heaters, etc.

This problem analyzes a fictitious vapor compression propane heat pump cycle utilizing a direct condensation radiator. This problem was chosen to demonstrate the following FLUINT features:

1. duct macros in a manifold/distribution design problem
2. compressor modeling using COMPRS
3. flow regime determination
4. slip flow modeling
5. duplication factors and dual one–way conductors for reduced model sizes
6. tolerance and elimination of junction loops

This problem is worked as both a detailed model and a simplified model. The detailed model focused on the compressor modeling, manifold design, and sensitivity to body force.

The simplified model demonstrates the simplification process itself, and also demonstrates various applications of slip flow modeling both by comparing homogeneous and slip flow assumptions, and by introducing a new component (a reflux tube) that can only be modeled using slip flow options.

G.1 Problem Description

The vapor compression propane heat pump depicted in Figure G–1 is used to acquire heat at –20°F, and transport it to a direct condensation radiator, where it is rejected to an effective sink of 0°F. The electrical input to the compressor is equivalent to 200 BTU/hr, and the compressor's efficiency is 80%. The payload source is represented by a fixed temperature, and the evaporator consists of a simple one–pass line.

The radiator is a thin 3'x3' aluminum sheet that views space on both sides (e.g., is deployed on orbit). A 0.080" vapor manifold is used, with eight evenly spaced parallel 0.035" condenser lines and a 0.060" liquid manifold. The vapor inlet is located at the corner opposite the liquid outlet for improved flow distribution. To return the liquid to the original corner (a convenient layout for deployment), and to simultaneously subcool the liquid (although not really necessary for this type of loop), the liquid manifold line is continued back across the radiator. The subcooler causes a sharp temperature gradient in the plate because of the relatively insignificant energy associated with sensible cooling compared to condensation—the rest of the radiator will be nearer the saturation point of 80°F. Another gradient will exist near the vapor end of the condenser lines because of the superheat anticipated. Thus, the maximum temperature gradient will occur in the lower left hand corner of radiator as shown in Figure G–1.

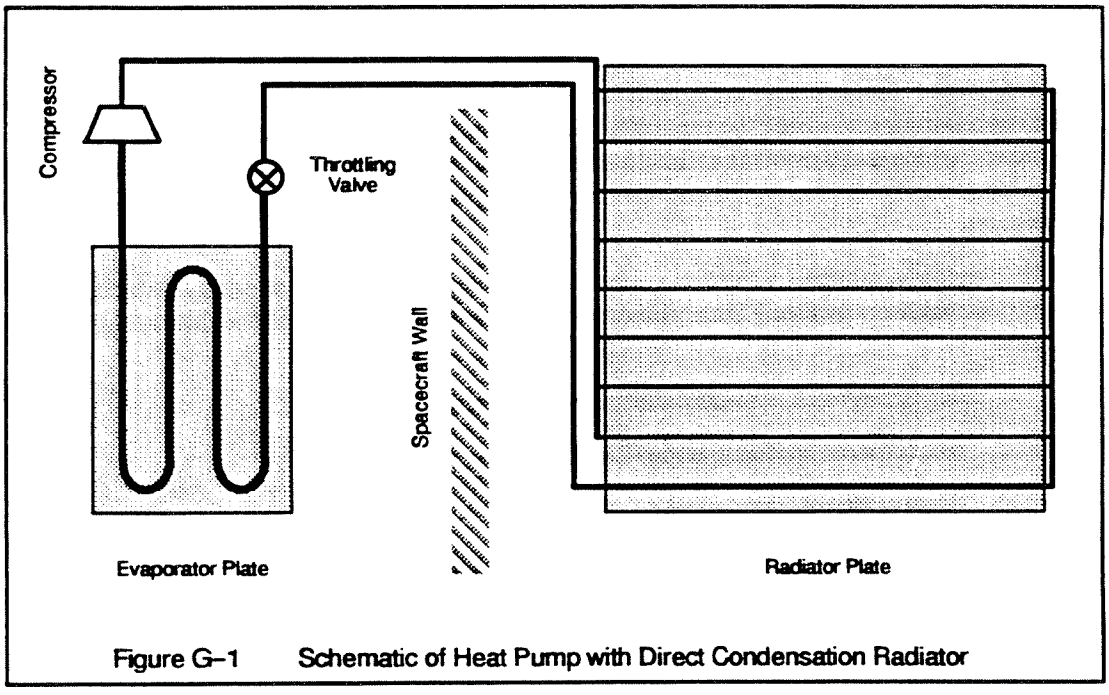


Figure G–1 Schematic of Heat Pump with Direct Condensation Radiator

The parameters for this propane heat pump system are listed below:

Compressor:	
Input power	200 BTU/hr
Isentropic efficiency	0.8
Outlet pressure	Saturation at 80°F (143.714 psia)
Radiator Plate:	
Length	3 ft
Width	3 ft
Thickness	0.010 inch
Conductivity	90 BTU/hr-ft-F
Radiator Plumbing:	
Vapor manifold ID	0.080 in
Condenser tube ID (except subcooler)	0.035 in
Liquid manifold ID	0.060 in
Subcooler and Liquid return line ID	0.060 in
Radiation Environment:	
Effective sink temperature	0°F
IR Emissivity	0.8
Number of active sides	2 (deployed)
Radiative fin effectiveness	0.86
Throttling Valve:	
ID	0.060 in
K Factor	20,000
Evaporator:	
Line length	6 ft
Line ID	0.125 in
Plate temperature	-20°F

One goal of the analysis is to verify adequate flow distribution in the condenser. The worse the distribution, the larger the radiator becomes to assure complete condensation. The problem becomes trivial as the manifold diameters increase, or as the condenser line diameter decreases, or both. However, the weight penalties associated with large manifolds and the pressure losses associated with small condenser lines require more design effort be expended. The velocities in the liquid lines will be much smaller than those in the vapor lines, requiring smaller diameters in the liquid manifold to match the pressure gradient in the vapor manifold. Furthermore, the effects of side–flow passages on the manifolds are not trivial. The vapor manifold will recover pressure as the axial velocity slows, while the pressure gradient (dP/dx) in the liquid manifold will increase as the velocity increases. Because the unit is intended for operation in zero gravity, but must be tested on ground in a horizontal position, the effects of gravity on the manifold performance should also be checked.

Another goal is to verify the overall operation of the system for the boundary conditions and power input provided. Assuming that the source and sink temperatures are not design parameters, this reduces to assuring that the compressor size and throttling valve setting are appropriate, and that the evaporator large enough to assure complete evaporation.

To preserve simplicity, this sample problem hides the process that resulted in the above design. Very few combinations of input compressor powers, boundary temperatures, throttling valve settings, or high–end pressures result in a sensible design much less a mathematically valid answer. For example, too much power for a fixed high–end pressure can cause incomplete condensation and/or the low–end pressure to drop down to where frictional resistances associated with high velocity (low density) vapor cause property calculation errors. Many modeling iterations were required even after a valid radiator design was found that evenly distributed flows with a reasonable pressure drop—and arriving at *that* design required many tries, not all of which were physically valid. Debugging the model *and* the design takes time and patience. It

is not easy to track down the true cause (or causes) of property error messages such as "PRESSURE TOO LOW" when the program is unable to arrive at even a consistent much less correct state. Start small, and verify component models independently if possible.

G.2 A SINDA/FLUINT Model

The compressor and throttling valve divide the loop into two pressure levels. The radiator is at a high temperature and pressure, and the evaporator is at a low pressure and temperature. Since the pressure at the high end of the heat pump is specified, attaching a plenum to the compressor outlet via a small STUBE connector will make sure the high end specification is met. The low pressure will then vary as needed to balance energy in and out of the heat pump—care must be taken to avoid unrealistically low pressures and densities, which will in turn cause even lower pressures at the evaporator exit due to frictional resistances.

The compressor can be modeled with a NULL connector whose input power is applied to the outlet lump. The simulation routine COMPRS can be called within FLOGIC 1 to take over simulation of the compressor. The adiabatic throttling valve is modeled with a simple LOSS connector.

The evaporator is only roughly defined. A single HX macro is used to model the one–pass evaporator, and the lumps are tied to a boundary node representing the source or payload.

The real focus of the model is the direct condensation radiator. To avoid a multiplicity of nodes that would be required to capture the gradients between condenser lines, a fin effectiveness method is used and only the root node is modeled. Conduction is modeled between adjacent root nodes on the basis of the total distance between their centers. This approximation has been shown to be very good even in such extreme conditions as zero flow in one line.

Because one of the goals is to verify manifold performance, the manifolds are not assumed to be perfect. Rather, LINE macros will be used to model the manifolds. In addition to variations in frictional gradient caused by axial velocity gradients, LINE macros will include the corresponding momentum flux gradients (i.e., the $\partial V^2/\partial x$ term). Such spatial accelerations due to side flow passages would not be included if individual lumps and paths had been input to represent the manifolds. For example, pressure recovery can occur in the vapor manifold, resulting in a U–shaped pressure profile. In the liquid manifold, the pressure gradient will be steeper when a LINE macro is used than it would if individual elements had been input, since fluid must be accelerated.

Faithful modeling of the manifolds forces modeling of individual condenser lines; duplication options cannot be used. (See simplified model.) An axial resolution of 9 segments is chosen. As a result, the plate is broken into 36 identical 4"x4" squares, as shown in Figure G–2, which also documents node and lump assignments. Strictly, loss factors should be included at the beginning and end of every condenser line to cover inlet, outlet, and turning losses. Such factors tend to result in more even distribution, so neglecting them should produce conservative results. On the other hand, carry–over in manifold tees is also neglected because it is difficult to characterize even though it does not promote even distribution. If such secondary loss terms need to be included, they can be supplied as FK factors on the first and last paths in each parallel HX macro.

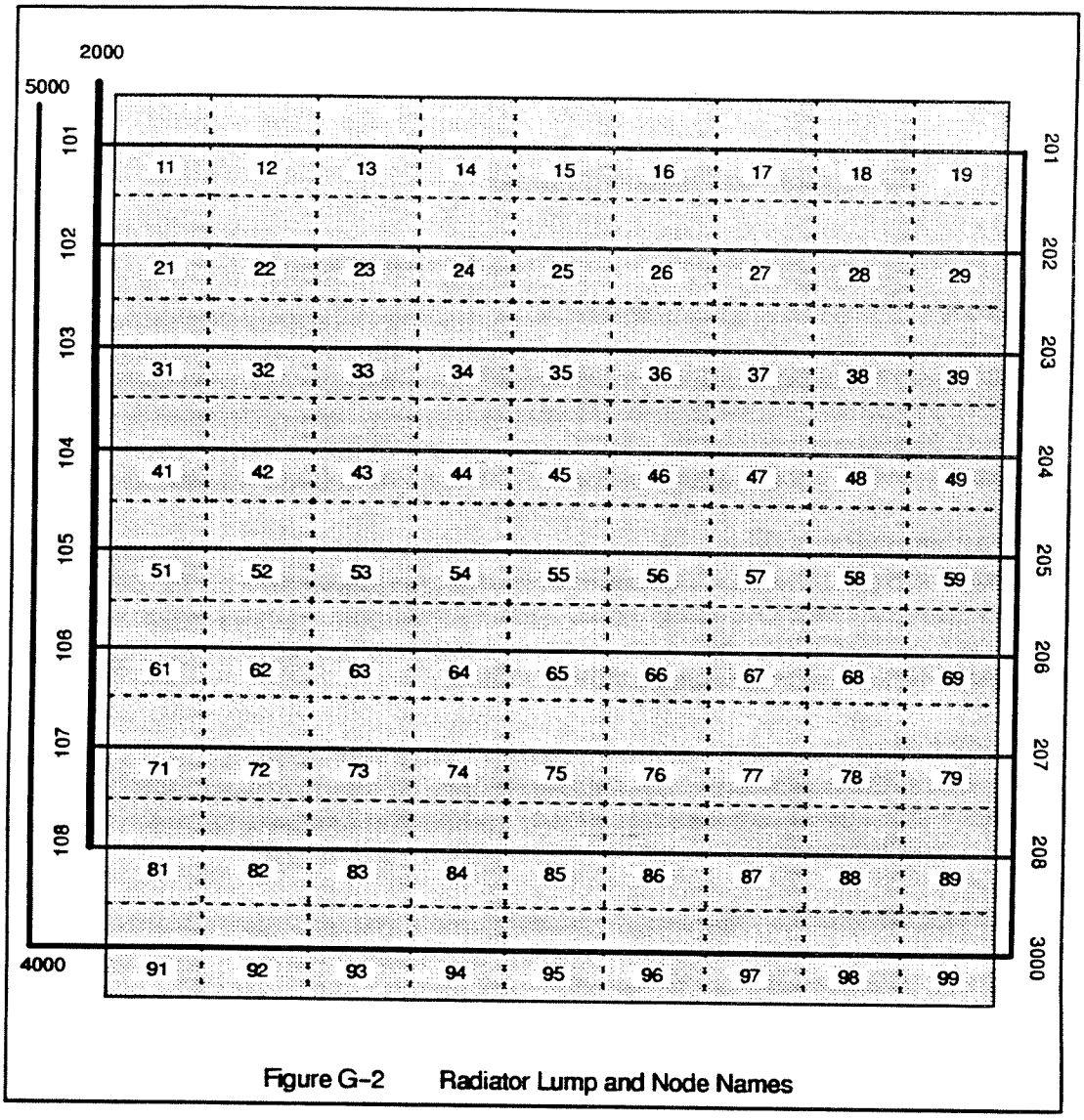


Figure G-2 Radiator Lump and Node Names

If IPDC=6 and the ACCEL vector magnitude is nonzero, SINDA/FLUINT will assume that all tubes and STUBEs are perpendicular to the ACCEL vector even if all lump coordinates are defaulted to CX=CY=CZ=0.0. The purpose of this treatment is to allow quick assessment of zero-g behavior versus one-g horizontal behavior. This feature will be exploited in this model to compare zero-g distribution with one-g distribution simply by turning one component of ACCEL on and off. Otherwise, the acceleration component imposes no force on any path. Rather, it only affects the flow regime determination, and therefore indirectly affects the local frictional gradient calculation.

The flow in all paths is assumed to be homogeneous. Slip flow paths, while easy to input, are more expensive to analyze, less likely to converge, etc. Furthermore, unless the void fraction changes enough to affect the flow regime determination, the results will be nearly identical to a

homogeneous analysis in terms of flowrates, pressure gradients, and heat transfer rates. For steady–state analyses, the only important difference will be void fraction predictions. Sensitivity to this assumption is tested in the simplified model, which is described later.

G.3 The Input File: Detailed Model

The input file is listed immediately following the last section.

Review of the following model reveals that a closed loop of diabatic junctions, or "junction loop," has been formed. As noted in Section 4.7.3.1, this is illegal in any solution routine except FASTIC since slight imbalances in energy flow can become trapped. Mathematically, if the energy flows within the loop are not perfectly balanced, no solution exists. Fortunately, this model uses tied junctions and only uses the FASTIC routine (see OPERATIONS DATA below). Actually, in many models junction loops are tolerated. Warning messages will be produced otherwise.

CONTROL DATA—English units are chosen, with units of °F and psia. An acceleration vector is applied in the Z direction. Because no elevations are specified later in FLOW DATA, this does not impose a direct force on any paths. Rather, it influences the flow regimes determined for all paths with IPDC=6, which are all treated as horizontal (perpendicular to any acceleration) for that purpose even though all CX=CY=CZ=0.0.

NODE DATA—The PLATE submodel contains 81 nodes each representing the center of 36 4"x4" squares. The numbering scheme for the radiator is depicted in Figure G–2. The effective radiation sink temperature is input as boundary node #999 at a temperature of 0°F. Also, one boundary node representing the payload (source) plate is input at –20°F.

CONDUCTOR DATA—Three sets of conductors are added to the PLATE submodel: (1) radiation to space (node 999), assuming two–sided radiation, an emissivity of 0.8, and a fin effectiveness of 0.86; (2) axial conduction between flow–wise centers; and (3) cross conduction between root nodes (perpendicular to the condenser lines).

FLOW DATA—One fluid submodel named COOL is used. The working fluid is "290," meaning the full library description of propane. The input starts at the compressor inlet (junction 1000), proceeds through the compressor, into the vapor manifold, across the radiator plate, through the liquid manifold, across the throttling valve, and through the evaporator. The names and types of the major elements are as follows:

Compressor:

JUNC 1000	compressor inlet
NULL 1000	for compressor simulation using call to COMPRS in FLOGIC 1
JUNC 2000	compressor outlet; attachment of reference plenum 1999

Radiator:

LINE 100	Vapor manifold: JUNC 101–108,STUBES 101–108
HX 11–81	Condenser lines. HX n1 extends from junction 10n to 20n, contains JUNC n1–n9, ties to nodes n1–n9, etc.
LINE 200	Liquid manifold: JUNC 201–208,STUBES 201–208
JUNC 3000	Outlet of liquid manifold, inlet of subcooler
HX 91	Subcooler, JUNC 99–91 (negative increment)
JUNC 4000	Outlet of subcooler, inlet to return line
STUBE 4000	Liquid return line

Throttling valve:

JUNC 5000	Outlet of return line, inlet of throttling valve
LOSS 5000	Throttling valve
JUNC 6000	Outlet of throttling valve, inlet to evaporator

Evaporator:

HX 1	JUNC 1-6, STUBE 1-7, all ties to PLATE.1

OUTPUT CALLS—This block calls standard output routines LMPTAB, TIETAB, and PTHTAB. Since all analyses are steady–states, it contains logic to avoid repeating initial conditions.

OPERATIONS DATA—The configuration is built, containing all input submodels. NLOOPS is set to 250, and FASTIC is called to find the first steady–state. NODMAP is called to produce detailed outputs for the portion of the radiator containing the largest gradients: node 91, where the exit of the subcooler is adjacent to the inlet of the last condenser leg. Finally, the acceleration vector is zeroed and another call to FASTIC is made.

FLOGIC 0—The COMPRS simulation routine is called from this location. To prevent liquid at the inlet of the compressor, a check is made for the quality of junction 1000. If any liquid is detected, the quality is reset to 1.0 and a warning is printed to the output file. As can be seen in the full output file, only one or two such corrections are made at the beginning of the run.

FLOGIC 2—The pressure profile in the manifolds is printed to the user file USER1 at the end of the steady state runs. These pressures are referenced to the lowest pressure such that differences can be distinguished. Note that this logic presumes a certain input order for the referenced lumps. Normally, this is a dangerous assumption which may make future changes difficult, and should be avoided in preference to use of INTLMP. However, because these lumps are generated by macros, the risk is minimal.

G.4 Output Description: Detailed Model

Selected printings from the OUTPUT file are provided on the pages following the input listings.

The low end pressure is about 14 psia, for a total pressure ratio of about 10. The quality at the outlet of the throttling valve is about 0.27, and the residual liquid is quickly evaporated in the first few feet of the evaporator, with the vapor nearly reaching the wall temperature by the time it exits the evaporator. Similarly, the superheat at the inlet of the radiator is quickly radiated in the first few inches, and the liquid subcools by about 30 degrees before entering the subcooler, where the subcooling is increased to about 40 degrees. While subcooling is slowed by the presence of an adjacent warm line, it does continue cooling along the entire length of the subcooler. (Reheating by the in–flowing vapor that is only four inches away is possible under other conditions.)

The principal measure of successful manifold performance is the distribution of flowrates between parallel lines, as shown in Figure G–3. If no subcooling line were attached, the outlets would all be at nearly the same temperature. However, the presence of the subcooling line causes a secondary gradient, which is also plotted in Figure G–3. This gradient also implies that

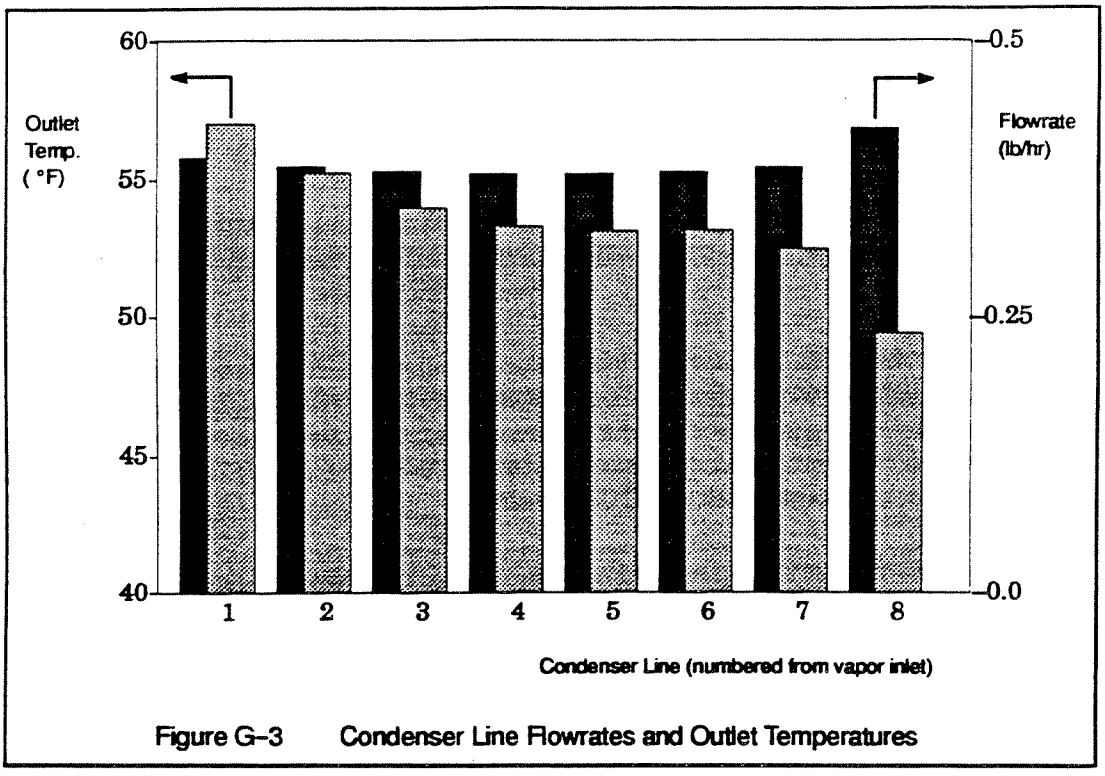


Figure G–3 Condenser Line Flowrates and Outlet Temperatures

a difference in resistance would exist between the parallel lines even if the flowrates were all equal since the lines nearer the subcooler contain more low–velocity liquid than those near the vapor inlet. Figure G–4 depicts the pressures in the manifold lines, illustrating the importance of the velocity gradients, and the resulting difficulties in making flows balance.

When ACCELZ is nonzero in the first run, a horizontal ground test is simulated. Almost all of the flow in the two–phase zone of the condenser is annular, with a brief occurrence of slug flow at the end of this zone. All of the flow in the evaporator is annular. If the mass fluxes were less and/or the gravity higher, the stratified regime would emerge in favor of the annular regime. At slower flows and reduced gravity, the bubbly regime would emerge in favor of the slug regime, although both regimes invoke the same homogeneous pressure loss model.

When ACCELZ is zeroed for the second run, no differences result. Because of the high mass fluxes and relatively high pressure gradients, the regime selection is insensitive to gravity. This insensitivity is normally a design goal because it maximizes confidence in ground test results. However, in many circulatory heat transport systems that are nearly isobaric, large pressure gradients and the corresponding insensitivity to gravity are a luxury that cannot often be afforded.

As a final note, because of flow regime hysteresis, some differences might result if the two FASTIC analyzes are reversed. For example, the bubbly regime may appear if the zero–g run were made first, then that regime might even persist in the subsequent one–g run if the distinction between slug and bubbly was within the range of uncertainties.

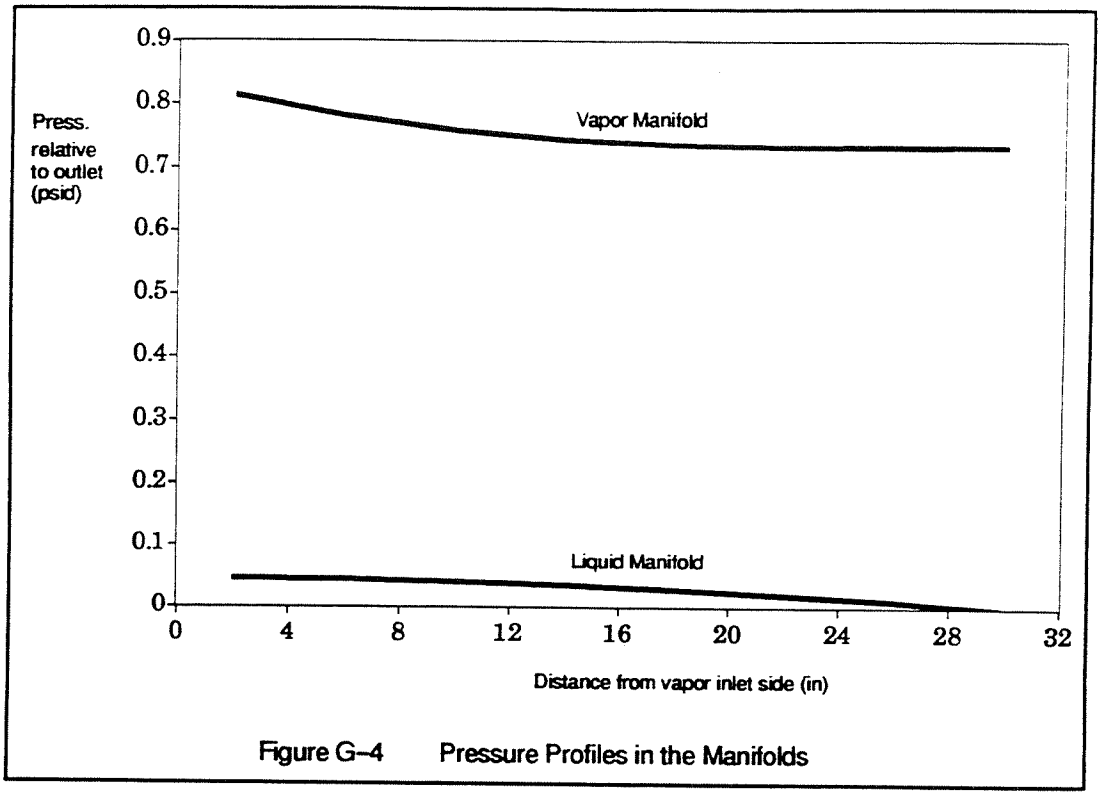


Figure G–4 Pressure Profiles in the Manifolds

G.5 Simplified Model for System Level Analyses

While a detailed model of the radiator is necessary for proper manifold design, the CPU costs associated with that level of detail are not appropriate for a vehicle–or system–level analysis. For such analyses, the emphasis is not on the performance of the condenser as much as on the macroscopic behavior of the heat pump as a subsystem. Any approximations are valid that would enable a simplified radiator model to mimic the overall resistance and heat transfer performance of the real system over a wide range of boundary conditions. In many ways, a detailed model of the radiator is *easier* to create than is a simplified model that executes faster while retaining the same basic behavior. While a detailed model requires more effort in terms of bookkeeping and typing, a simplified model requires more engineering and SINDA/FLUINT expertise, and more creativity.

As will be demonstrated, the simplified model produces nearly identical results at a fraction of the computational cost of the detailed model. Perhaps more importantly, the results are easier to interpret.

A very top–level representation of the heat pump might avoid fluid submodels altogether, using a few SINDA nodes and conductors with composite temperatures, average heat transfer coefficients, and an effective coefficient of performance or percent of Carnot efficiency, etc. The model that follows is aimed at a more intermediate level of detail, where the important temperature, pressure, and quality gradients in the heat pump and its components are analyzed.

Compressor and valve—The compressor and throttling valve are already as simple as can be while still employing a fluid submodel representation of the heat pump. The focus is therefore on the simplifications that can be made in the evaporator and radiator models.

Evaporator—The pressure gradients in the evaporator (about 1.3 psid) are somewhat large, especially in comparison to the absolute pressure (about 14 psia). However, this gradient is not very important from a system perspective because of the intentionally dominant resistance of the throttling valve, and because of the absence of elements in parallel with the evaporator, which would force the use of good resistance model because of flow distribution. Also, the evaporator has relatively high effectiveness as a heat exchanger, with only about a degree difference between outlet and wall temperatures.

Taking this performance into account, a very simplified model of the evaporator might be a single STUBE with all heat flow occurring in a single junction upstream of the STUBE. Either the junction QDOT or its enthalpy (using HTRLMP) could be calculated using a simple heat exchanger effectiveness method. The resistance of an STUBE carrying 100% vapor would be close enough to that of the real evaporator, which has two–phase flow in the first third of its length. The payoff for such an approach, in terms of the number of elements saved, is not very great compared to the reduced ability to simulate off–design cases.

In this model, the number of segments is simply reduced from six to four. Use of even fewer elements tends to underpredict the heat transfer in the single–phase zone (refer to Section 4.7.5). While HTNS ties (employed by the DS option in HX macros) overcome this problem, they are not as appropriate for the two–phase heat transfer that occurs at the inlet of the evaporator, and in fact revert to HTN ties in such instances.

Either method represents a compromise; the user is encouraged to experiment with both. An ideal model would use an adequate number of lumped parameter ties (HTN and HTNC) in the two–phase portion, then model the single–phase portion with a single HTNS segment. However, since the lengths of the two–phase and single–phase regions are variable, a compromise solution is used in this model: HTNC ties with the minimum number adequate to resolve the gradients.

Radiator—A parallel reduction is made in the radiator model, reducing the number of axial segments from nine to six. Again, the above discussion of segment–oriented single–phase vs. lumped two–phase heat transfer options is relevant here.

Because of the number of parallel segments, this change alone eliminates the need for 9x3=27 lumps and corresponding nodes. However, if a means can be found to eliminate separate models for each parallel condenser leg, up to 7x6=42 more lumps can be eliminated.

If the manifolds ideally perform their intended function, then the flowrates in each leg can be assumed equal and duplication factors can be used to model a single line, eliminating the aforementioned 42 lumps. Each manifold will be modeled with a single junction and STUBE,

with one HX macro (using DUP=8.0) that extends from the vapor manifold junction to the liquid manifold junction, which represent average pressure conditions in those manifolds. Assuming quasi–uniform extraction and injection in the manifolds, half–length modeling can be applied (Section 4.7.12). The manifold STUBEs are therefore input as one–half of the true lengths, and AC factors are calculated in FLOGIC 0 to account for the side–flow passages.

Unfortunately, the radiator itself is not thermally symmetric because of the existence of the subcooler on one end. If it were symmetric, then it could be reduced to a single row of nodes representing a 1D gradient (the secondary gradient perpendicular to the pipes having been taken into account with a root node and fin effectiveness method).

To compromise, the thermal model could be left as–is, yet the fluid model can still be DUPed to reduce lumps. This method exploits the fact that nodes are much less expensive than lumps. In this case, the six lumps in the single HX macro can be tied to all nodes in the radiator. One row (perhaps nodes 11–16) could be created in the HX macro itself, and then seven additional rows of ties could be generated with seven calls to GENT using the HXC option. For all these ties, DUPL=1.0/8.0 would be input, and DUPN=1.0 is defaulted. In other words, the heat transfer into each lump would be averaged on the basis of the distinct temperatures at each of eight nodes located at the same axial coordinate. From the perspective of each of these nodes, the full lump heat rate is applied to that node, and does not appear to be shared amongst other nodes.

As can be seen in the above discussion, *the whole topic of simplification relies heavily on creating symmetries by assumption, and then exploiting them by duplication.* The duplication process creates asymmetries in the network: the elements on either end of a path, tie, or conductor may not be in agreement over the size (or number) of that path, tie, or conductor. An in–depth understanding of this process yields another, more subtle network reduction: the number of nodes *can* be reduced without compromising the importance of the subcooler–induced temperature gradient using dual, opposing one–way conductors (Section 3.10).

Again, a symmetry must be created or assumed in order to be exploited. In this case, the new assumption is the radiator temperatures next to the condenser lines can be modeled with a single row of nodes representing average conditions. From the viewpoint of the subcooler, the nodes representing such an average pipe wall condition should be immediately adjacent to it. However, the line adjacent to the subcooler does not itself represent a good average state, whereas a line in the middle of the plate would be a much better representation. This difference in viewpoint is accommodated by changing a single row of normal conductors representing cross conduction into two rows of opposing one–way conductors. From the subcooler's point of view (as the downstream node), the nodes representing an average condenser line appear to be only 4" away. However, the nodes representing that average state believe they are 18.5" away from the subcooler, which corresponds to the center for the original eight condenser lines.

The final radiator model for the simplified input is shown in Figure G–5.

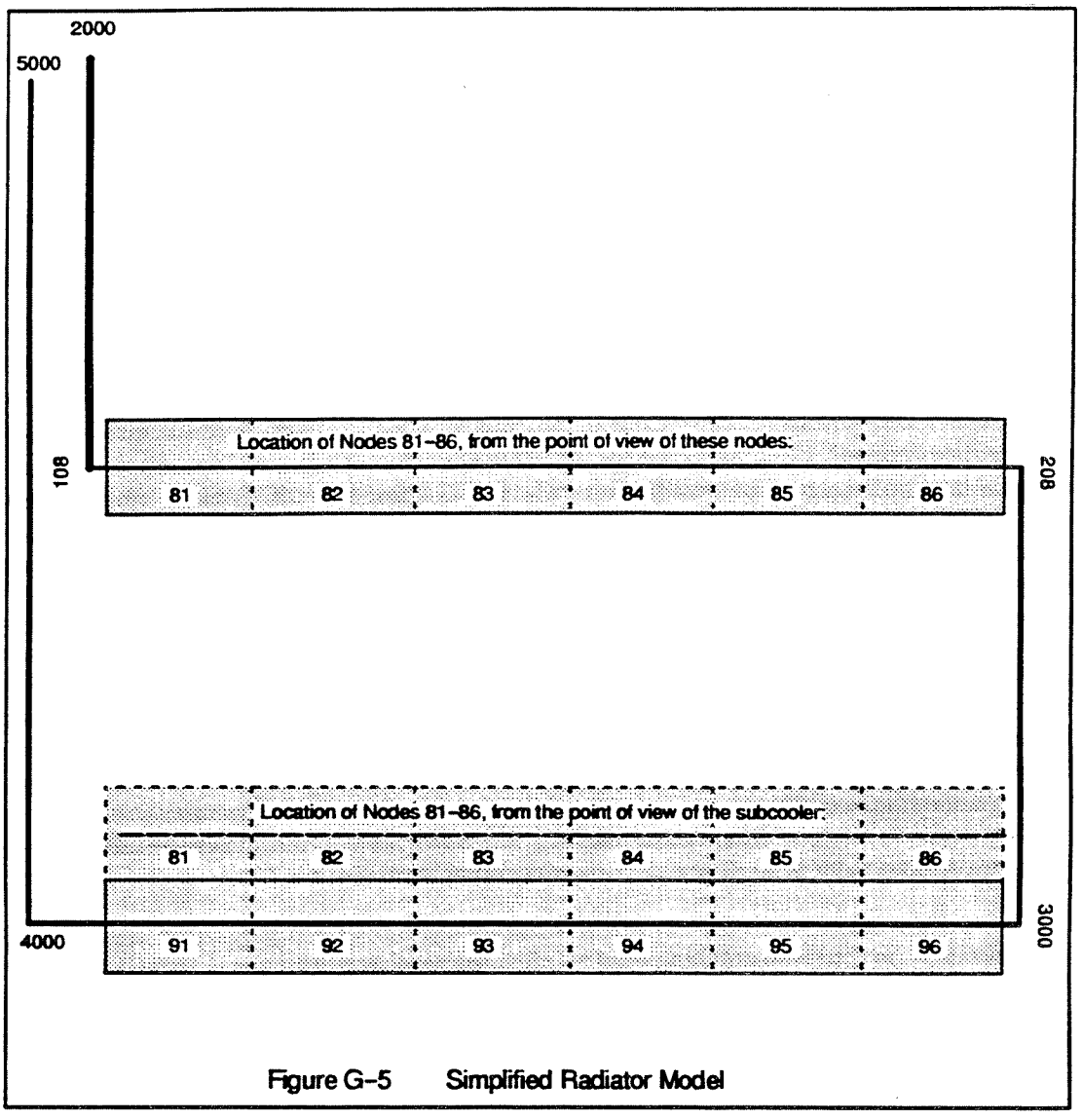


Figure G-5 Simplified Radiator Model

Fluid description—A final simplification would be to use a faster, more approximate fluid description using RAPPR. Unfortunately, unlike the CPL sample problem (Problem E), this easy step is much less appropriate because of the large range of pressures required. A RAPPR description covering the same region would require accepting the default errors of 10% in both liquid and vapor properties using an average (reference) saturation state of 25°F. This figure represents the error in the final results, which are all near the borders of this valid range. The actual RAPPR description would have to have much more maneuvering room (i.e., error tolerance) to avoid hitting property limits while a valid steady state solution is being sought. (Normally, the actual errors in the results are much smaller and quite acceptable for most systems with a narrower range of operating pressures. Use of RAPPR is strongly recommended when possible.)

G.6 Pre–flight Noncondensible Gas Trapping: Reflux Tube

Before the heat pump unit is flown, it is desired to collect and purge any noncondensible gases (NCG) in the loop that were generated by residual contaminants or were themselves the residuals of an imperfect pre–charge vacuum. One method of trapping such gases is to condense a portion of the fluid in a vertical tube attached to the vapor side of the heat pump. While the heat pump is running on ground, vapor and NCG will enter the tube. The vapor will condense into liquid that flows back out of the tube and into the loop because of gravity. The NCG will remain at the top of the tube and will be evidenced by a temperature gradient in that zone since no condensation is occurring. A purge valve can be used to bleed off the NCG until the reflux tube returns to an isothermal state. (Heat pipe and thermosyphon enthusiasts will recognize this device as the top half of a simple thermosyphon.) The longer the system is run, and the greater the proportion of condensation in the reflux tube compared to the condenser, the more NCG will be extracted prior to flight.

The parameters for this "reflux tube" are listed below along with additional information on the remainder of the heat pump:

Reflux Tube:

Length	2.5 ft
ID	0.25 in
OD	0.30 in
Conductivity	90 BTU/hr–ft–F
Ambient air temperature	70°F
Natural convection coefficient	2.0 BTU/hr–ft^2–F

System Volumes (excluding Reflux Tube)

Vapor volume	0.01 ft^3
Liquid volume	0.001 ft^3

The simplified model will be used to investigate the reflux tube and its effects on the rest of the system. To avoid an unnecessary departure from the intent of this example, the case with zero NCG will be considered.

Countercurrent flow in the reflux tube virtually requires the use of twinned paths (Section 4.7.14). Unfortunately, because of the closed end, the net flow is zero and junctions are therefore inappropriate. The requirement to use of tanks to model the reflux tube has several important ramifications on the rest of the model. First, the use of tanks in two–phase models implies the use of tubes as well to avoid instabilities. Therefore, tubes will be required in the reflux tube model.

Second, use of STDSTL will be required since FASTIC would treat the new tanks as junctions, defeating the reasons for their introduction. The remainder of the steady–state oriented model could still be built with junctions and STUBE connectors, except that the existence of the aforementioned junction loop (Section 4.7.3.1) can cause problems in routines other than FASTIC. (Actually, the occurrence of failures due to the presence of junction loops is rare and sporadic. Junction loops will be avoided nonetheless.) Furthermore, the volume of the reflux tube is small compared to that of the remainder of the loop, and modeling the volume of the reflux tube (by using tanks) while neglecting the volume of the rest of the system (by using junctions) doesn't make sense. For these reasons, two of the junctions in the model will be converted to tanks. Tank 1000 (compressor inlet) will contain the total system vapor volume, and Tank 5000 (condenser outlet) will contain the total system liquid volume.

Despite the use of these two tanks, the two parts of the model are still somewhat mismatched. The tanks in the reflux tube model may be driven unstable by their attachment to an instantaneous junction. To compromise, the two models are run independently, and then combined into a final solution. First, FASTIC is called, which results in zero flow in the reflux tube because of the treatment of tanks as junctions in that routine. This run therefore simulates the simplified heat pump without a reflux tube. Next, the reflux tube is reinitialized, the compressor outlet is held using HLDLMP to minimize interactions, the reservoir (plenum) is removed from the system via the use of zero path DUP factors, and STDSTL is called. This second run solves for the flows and heat rates in the reflux tube with no changes in the rest of the heat pump because of the HLDLMP call. Finally, the HLDLMP is released using RELLMP, and STDSTL is again called to get a final combined solution.

The condensation heat transfer within the reflux tube deserves some thought. The standard convection heat transfer subroutines would underpredict the heat transfer across a thin, slow moving annular film. Instead of HTN ties, HTU ties will be added with a rough–guess condensation heat transfer coefficient: 50% of the conductance of the liquid film.

In summary, the reflux tube will therefore be modeled with a LINE macro built with tanks and tubes and connected to an arithmetic node wall model via HTU ties.

Because slip flow is used in the reflux tube, the sensitivity to the assumption of homogeneity in the rest of the system will be tested in the same model. The model will be input using twinned STUBEs in the condenser and evaporator. After a slip flow solution is found, the GOHOMO routine will be used to turn off slip flow in all but the reflux tube, which will enable a comparison with the detailed model. Two FASTIC calls will therefore be made to enable this comparison.

G.7 The Input File: Simplified Model with Reflux Tube

The input file for the simplified radiator model, which includes the reflux tube model, is provided at the end of this section after the input for the detailed model. Only changes from the detailed model are described.

NODE DATA—Only the last two rows of the radiator PLATE submodel are input, and the resolution has been made more coarse, leaving 12 nodes representing 6"x4" sections. A new submodel VERT has been added with five nodes representing the five vertical segments of the reflux tube. A new boundary node representing an on–ground ambient air temperature of 70°F has also been added.

CONDUCTOR DATA—The number of conductors has been greatly reduced from that used in the detailed model, and the values have been changed to reflect the coarser axial nodalization. Also, instead of one row of normal two–way conductors representing cross conduction, two rows of opposing one–way conductors have been input. This makes the subcooler view the remaining condenser line as adjacent, while the remaining condenser line views the subcooler more remotely—as if it existed in the middle of the plate, and is therefore more representative of the average.

In the new VERT submodel, two rows of conductors have been added representing axial conduction and the natural convection to ambient (through–thickness gradients being negligible).

FLOW DATA—One fluid submodel named COOL is used. The working fluid is "290," meaning the full library description of propane—a RAPPR subset is inappropriate because of the large pressure variation. The input starts at the compressor inlet (Tank junction 1000), proceeds through the compressor, side tracks into the reflux tube, and then returns to the input pattern followed in the detailed model:

Compressor:	
TANK 1000	compressor inlet
NULL 1000	for compressor simulation using call to COMPRS in FLOGIC 1
JUNC 2000	compressor outlet; attachment of reference plenum 1999
Reflux tube:	
LINE 2001	"uphill discretized" tanks 2001–2005 and twinned tubes where the primaries are 2001–2005, secondaries 2011–2015
HTU 2001–2005	annular film thickness condensation ties
Radiator:	
STUBE 108, JUNC 108	Vapor manifold
HX 81	Duped, twinned condenser lines, JUNC 81–86
STUBE 208, JUNC 208	Liquid manifold
JUNC 3000	Outlet of liquid manifold, inlet of subcooler
HX 91	Subcooler, JUNC 96–91 (negative increment)
JUNC 4000	Outlet of subcooler, inlet to return line
STUBE 4000	Liquid return line
Throttling valve:	
TANK 5000	Outlet of return line, inlet of throttling valve
LOSS 5000	Throttling valve
JUNC 6000	Outlet of throttling valve, inlet to evaporator
Evaporator:	
HX 1	JUNC 1–4, twinned primary STUBEs 1–5, all ties to PLATE.1

Note that the spatial resolution has been decreased 33% in both the condenser and evaporator, and that paths in both of these components have been twinned to include slip flow modeling.

OUTPUT CALLS—This block calls standard output routines LMPTAB, TIETAB, and PTHTAB. A call to TWNTAB has been added for those cases where twins are active. Since all analyses are steady–states, this block contains logic to avoid repeating initial conditions.

OPERATIONS DATA—The configuration is built, containing all input submodels. FASTIC is called to find the first steady–state, which results in zero flow in the reflux tube because of the treatment of tanks as junctions in that routine. This run therefore simulates the simplified heat pump without a reflux tube, but includes the effects of slip flow in the condenser and evaporator. Next, all paths are converted into the homogeneous mode by calls to GOHOMO. A second call to FASTIC is made.

Next, the reflux tube is reinitialized, slip flow is enabled in that device via a call to GOSLIP, the compressor outlet is held using HLDLMP to minimize interactions, the reference pressure is removed from the system by setting the DUPI factor on the connecting path (number 1999) to zero, and STDSTL is called. This third analysis solves for the flows and heat rates in the reflux tube with no changes in the rest of the heat pump because of the HLDLMP call. Finally, the HLDLMP is released using RELLMP, and STDSTL is again called to get a final combined solution.

Before completing the run, calls to LMXTAB and TUBTAB are made to print out extra data concerning the final state. Calls to these two routines are often more appropriately made in OPERATIONS DATA than in OUTPUT CALLS since the information they convey (e.g., lump coordinates, path lengths and diameters) is relatively invariant.

FLOGIC 0—First, axial velocity gradients that exist in the real manifolds are input as AC factors for the simplified one–segment model, assuming uniform extraction and injection. Second, the same COMPRS logic used in the detailed model is repeated. Third, if a routine other than FASTIC is being used, then crude heat transfer calculations are performed to estimate the rate of condensation in the reflux tube as a function of film thickness.

G.8 Output Description: Simplified Model

Selected printings from the OUTPUT file are provided immediately following the selected outputs from the detailed model.

In the first FASTIC run, the reflux tube is shut down in the resulting answer because of the treatment of tanks as junctions combined with the inability of junctions to accommodate counterflow cases with zero net flow. If the reflux tube had been absent altogether, the same answer would have been found at a fraction of the cost.

In the second run, the twinned paths have all been 'homogenized.' This run can then be compared with both the previous run to measure the importance of the slip flow assumption, and with the detailed model to measure the success of the simplifications.

Comparing with the previous answer, the results are nearly identical except for differences in the void fractions in the evaporator and condenser. Otherwise, pressure drops and heat transfer rates are essentially the same. Note that the "XL FLOW" column in the TWNTAB print out is nearly identical for both runs, and that these values also correspond to the lump qualities in the LMPTAB of the second run and the "XL UPSTRM" column in the PTHTAB of that same run. Unless differences in void fractions are required and are expected to be significant for such a steady–state run, there is no reason to bother with twinned paths. The only exception to this statement is the indirect effect that slip flow can have on flow regime and therefore pressure gradient. For example, pressure gradients should be considered suspect for stratified regimes when homogeneous flow is assumed because of the large error in void fraction estimation for that regime, and the sensitivity of the pressure drop calculation for stratified flow on that same void fraction.

Comparisons with the detailed model are excellent. All temperatures are within about one degree of each other, and those differences are attributable to the reduced resolution in the evaporator and condenser. A model with the same axial resolution as the full model (six lumps in the evaporator, nine in the condenser) achieves nearly identical results as the full model. The warning "STABLE BUT NOT BALANCED" is caused by the use of unequal opposing one–way conductors, and is of no consequence, as justified by the close match between results.

With respect to modeling the reflux tube, there is no choice between homogeneous and slip modeling: countercurrent flow inherently eliminates the homogeneous option. After the reflux tube has been reinitialized from its quiescent state (to minimize CPU time caused by starting from an unrealistic zero flow condition and potentially unrealistic lump qualities), and after lump 2000 has been held using HLDLMP to isolate the two halves of the system, a third steady–state run is invoked using STDSTL. The solution for the reflux tube is found in this intermediate run.

Finally, the call to HLDLMP is released using RELLMP, and the final combined result is found. The total rate of condensation in the reflux tube is about 4 BTU/hr, so less than 1% of the working fluid is recirculating through the reflux tube. Still, this has the effect of cooling the

compressor outlet by about 3°F. Within the reflux tube itself, the liquid film thickness is very thin at the top of the tube, then grows gradually as liquid flows downward and additional vapor condenses out onto the film. The heat transfer coefficient also varies according to the film thickness, but it is so large compared to other conductances in the system that it has no effect on the heat rate, which is nearly constant along the length of the reflux tube. No improvements to the first–order heat transfer logic appear to be warranted.

The flow in the reflux is completely annular, as anticipated by the heat transfer logic. If the pipe had slanted instead of being perfectly vertical, the stratified regime would have resulted. If instead the pipe diameter had been reduced, the slug regime would have formed at the bottom, perhaps causing intermittent behavior. Setting RSTUBF higher than the default of 0.5 help alleviate the ensuing small time steps, although the focus of the analysis would inevitably shift to the reflux tube, probably inappropriately.

Input File – Detailed Model

```
HEADER OPTIONS DATA
C FLUINT SAMPLE PROBLEM G
TITLE DETAILED REPRESENTATION OF DIRECT COND. HEAT PUMP RADIATOR
       OUTPUT = radiator.out
HEADER CONTROL DATA, GLOBAL
       UID    = ENG
       SIGMA  = .1713E-8
C ACCELERATION AFFECTS ONLY FLOW REGIMES
       ACCELZ = 32.2*3600.*3600.
C
C RADIATOR PLATE IS ALUM 36" x 36" x 0.010" THICK
C NODALIZE INTO 9 x 9 GRID OF 4" x 4" SQUARES
C
HEADER NODE DATA, PLATE
       GEN,11,9,1,70.0,-1.0
       GEN,21,9,1,70.0,-1.0
       GEN,31,9,1,70.0,-1.0
       GEN,41,9,1,70.0,-1.0
       GEN,51,9,1,70.0,-1.0
       GEN,61,9,1,70.0,-1.0
       GEN,71,9,1,70.0,-1.0
       GEN,81,9,1,70.0,-1.0
       GEN,91,9,1,70.0,-1.0
       -999,0.0,0.0
C
C SOURCE
C
       -1,-20.0,0.0
C
```

```
HEADER CONDUCTOR DATA, PLATE
C RADIATION TO SPACE
       GEN,-911,9,1,11,1,999,0, 2.*16./144.*0.8*0.86
       GEN,-921,9,1,21,1,999,0, 2.*16./144.*0.8*0.86
       GEN,-931,9,1,31,1,999,0, 2.*16./144.*0.8*0.86
       GEN,-941,9,1,41,1,999,0, 2.*16./144.*0.8*0.86
       GEN,-951,9,1,51,1,999,0, 2.*16./144.*0.8*0.86
       GEN,-961,9,1,61,1,999,0, 2.*16./144.*0.8*0.86
       GEN,-971,9,1,71,1,999,0, 2.*16./144.*0.8*0.86
       GEN,-981,9,1,81,1,999,0, 2.*16./144.*0.8*0.86
       GEN,-991,9,1,91,1,999,0, 2.*16./144.*0.8*0.86
C AXIAL CONDUCTION
       GEN,11,8,1,11,1,12,1, 90.0*0.010*4./4. /12.
       GEN,21,8,1,21,1,22,1, 90.0*0.010*4./4. /12.
       GEN,31,8,1,31,1,32,1, 90.0*0.010*4./4. /12.
       GEN,41,8,1,41,1,42,1, 90.0*0.010*4./4. /12.
       GEN,51,8,1,51,1,52,1, 90.0*0.010*4./4. /12.
       GEN,61,8,1,61,1,62,1, 90.0*0.010*4./4. /12.
       GEN,71,8,1,71,1,72,1, 90.0*0.010*4./4. /12.
       GEN,81,8,1,81,1,82,1, 90.0*0.010*4./4. /12.
       GEN,91,8,1,91,1,92,1, 90.0*0.010*4./4. /12.
C CROSS CONDUCTION
       GEN,201,9,1,21,1,11,1,       90.0*0.010*4./4. /12.
       GEN,301,9,1,21,1,31,1,       90.0*0.010*4./4. /12.
       GEN,401,9,1,41,1,31,1,       90.0*0.010*4./4. /12.
       GEN,501,9,1,41,1,51,1,       90.0*0.010*4./4. /12.
       GEN,601,9,1,61,1,51,1,       90.0*0.010*4./4. /12.
       GEN,701,9,1,61,1,71,1,       90.0*0.010*4./4. /12.
       GEN,801,9,1,81,1,71,1,       90.0*0.010*4./4. /12.
       GEN,901,9,1,81,1,91,1,       90.0*0.010*4./4. /12.
C
```

```
HEADER FLOW DATA, COOL, FID=290
C
LU DEF, TL = 80.0,     XL      = 1.0, PL      = 143.714
PA DEF, FR = 2.0,      IPDC    = 6
C
LU JUNC,1000,  PL! =15.0,      TL      = -30.0
C
C USE NULL DEVICE: WILL SIMULATE COMPRESSOR
C
PA CONN,1000,1000,2000, GK = 1.0     $ FAKE TO SATISFY INPUT REQMNTS
       DEV     = NULL
C
C DEFINE TOP END: ATTACH REFERENCE POINT
C
LU PLEN,1999
PA CONN,1999,1999,2000
       DEV     = STUBE
       DH      = 0.02/12.0
       TLEN    = 1.0
C
LU JUNC,2000,  QDOT    = 200.0
C
C VAPOR HEADER
C
M LINE,100,D,101,101,2000,    NSEG    = 8
       DHS     = 0.080/12.0,  TLENT   = 2.7
       PA      = STUBE,       LU      = JUNC
C
C RADIATOR PIPES
LU DEF, XL = 0.1,      TL = 80.0
PA DEF, FR = 0.25
C
M HX,11,C,11,11,11,PLATE.11,101,201,        NSEG    = 9
       DHS     = 0.035/12.0,  TLENT   = 3.0
       PA      = STUBE,       LU      = JUNC
M HX,21,C,21,21,21,PLATE.21,102,202,        NSEG    = 9
       DHS     = 0.035/12.0,  TLENT   = 3.0
       PA      = STUBE,       LU      = JUNC
M HX,31,C,31,31,31,PLATE.31,103,203,        NSEG    = 9
       DHS     = 0.035/12.0,  TLENT   = 3.0
       PA      = STUBE,       LU      = JUNC
M HX,41,C,41,41,41,PLATE.41,104,204,        NSEG    = 9
       DHS     = 0.035/12.0,  TLENT   = 3.0
       PA      = STUBE,       LU      = JUNC
M HX,51,C,51,51,51,PLATE.51,105,205,        NSEG    = 9
       DHS     = 0.035/12.0,  TLENT   = 3.0
       PA      = STUBE,       LU      = JUNC
```

```
M HX,61,C,61,61,61,PLATE.61,106,206,        NSEG    = 9
       DHS     = 0.035/12.0,  TLENT   = 3.0
       PA      = STUBE,       LU      = JUNC
M HX,71,C,71,71,71,PLATE.71,107,207,        NSEG    = 9
       DHS     = 0.035/12.0,  TLENT   = 3.0
       PA      = STUBE,       LU      = JUNC
M HX,81,C,81,81,81,PLATE.81,108,208,        NSEG    = 9
       DHS     = 0.035/12.0,  TLENT   = 3.0
       PA      = STUBE,       LU      = JUNC
C
C LIQUID HEADER
LU DEF, XL = 0.0,     PL = 0.0
C
M LINE,200,U,201,201,3000,    NSEG    = 8
       DHS     = 0.060/12.0,  TLENT   = 2.7
       PA      = STUBE,       LU      = JUNC
       FR      = 2.0
C
LU JUNC,3000
LU JUNC,4000
C
PA DEF,FR      = 4.0
       DH      = 0.060/12.0
C
C SUBCOOLER
C
M HX,91,C,99,100,99,PLATE.99,3000,4000,     NSEG    = 9
       LUINC   = -1,          PAINC   = -1,
       DHS     = 0.060/12.0,  TLENT   = 3.0
       PA      = STUBE,       LU      = JUNC
       NINC    = -1
C
PA CONN,4000,4000,5000
       DEV     = STUBE
       DH      = 0.060/12.0,  TLEN    = 3.0
C
LU JUNC,5000,  XL      = 0.0
C
C THROTTLING VALVE
C
PA CONN,5000,5000,6000,
       DEV     = LOSS
       FK      = 20000.0
C
LU DEF,XL      = 0.1, PL!     = 15.0,TL      = -30.0
LU JUNC,6000
C
C EVAPORATOR: SINGLE PASS
```

```
C
M HX,1,C,1,1,1,PLATE.1,6000,1000,    NSEG    = 6
        NINC    = 0
        DHS     = 0.125/12.0,  TLENT   = 6.0
        PA      = STUBE,       LU      = JUNC
        XL      = 0.3,         XLINC   = 0.1
C
HEADER OUTPUT CALLS, COOL
        IF(LOOPCT .EQ. 0)RETURN
        CALL LMPTAB('ALL')
        CALL TIETAB('ALL')
        CALL PTHTAB('ALL')
C
HEADER OPERATIONS DATA
BUILD RAD, PLATE
BUILDF RAD, COOL
        NLOOPS          = 250
DEFMOD COOL
C
        CALL FASTIC
C
C CHECK CROSS CONDUCTION IN LAST PART OF CONDENSER
C
        CALL NODMAP('PLATE',91,0)
C
C NOW ELIMINATE BODY FORCE AND REPEAT
C
        LOOPCT          = 0
        ACCELZ          = 0.0
        CALL FASTIC
C
```

```
HEADER FLOGIC 0, COOL
C
C SIMULATE COMPRESSOR:
C
       IF(XL1000 .LT. 1.0)THEN
              WRITE(NOUT,10) XL1000,LOOPCT
10            FORMAT(' CHANGED QUAL FROM ',F8.6
     .                ,' AT LOOPCT = ',I4)
              CALL CHGLMP('COOL',1000,'XL',1.0,'PL')
       ENDIF
       CALL COMPRS('COOL',1000,0.8)
C
HEADER FLOGIC 2, COOL
C
C PRINT MANIFOLD PRESSURE PROFILES RELATIVE TO LOWEST PRESSURE
C (WHICH IS LIQUID EXIT, LUMP 208).
C
       WRITE(NUSER1,1) SNGL(PL208)
       WRITE(NUSER1,10)(SNGL(PL(101+ITEST)-PL208),ITEST=0,7)
       WRITE(NUSER1,10)(SNGL(PL(201+ITEST)-PL208),ITEST=0,7)
1      FORMAT(//' RELATIVE PRESSURES IN MANIFOLDS, LINES 1 THROUGH 8,'
     .  ' RELATIVE TO ',1PG15.5,' PSIA '/)
10     FORMAT(1X,8F10.4)
```

Input File – Simplified Model

```
C ===========================================================================
C BEGIN INPUT FILE FOR SECOND HALF: REDUCED MODEL WITH REFLUX TUBE
C ===========================================================================
C
HEADER OPTIONS DATA
TITLE SYSTEM REPRESENTATION OF DIRECT COND. HEAT PUMP RADIATOR
C ALSO HAS REFLUX TUBE, SLIP FLOW SENSITIVITY
       OUTPUT = sysflux.out
HEADER CONTROL DATA, GLOBAL
       UID    = ENG
       SIGMA  = .1713E-8
       RSMAXF = 1.0E-2
       NLOOPS = 250
C THIS AFFECTS FLOW REGIMES ONLY IN RADIATOR AND EVAPORATOR
       ACCELZ = 32.2*3600.*3600.
C
C RADIATOR PLATE IS ALUM 36" x 36" x 0.010" THICK
C NODALIZE INTO 6 x 9 GRID OF 6" x 4" SQUARES
C FIRST 8 ROWS ARE ASSUMED IDENTICAL
C
HEADER NODE DATA, PLATE
       GEN,81,6,1,70.0,-1.0
       GEN,91,6,1,70.0,-1.0
       -999,0.0,0.0
C
       -1,-20.0,0.0
C
HEADER CONDUCTOR DATA, PLATE
C RADIATION TO SPACE
       GEN,-981,6,1,81,1,999,0, 2.*24./144.*0.8*0.86
       GEN,-991,6,1,91,1,999,0, 2.*24./144.*0.8*0.86
C AXIAL CONDUCTION
       GEN,81,5,1,81,1,82,1, 90.0*0.010*4./6. /12.
       GEN,91,5,1,91,1,92,1, 90.0*0.010*4./6. /12.
C CROSS CONDUCTION
C TRICKY! 8TH ROW APPEARS NEXT TO 9TH ROW, BUT
C         9TH ROW APPEARS FARTHER AWAY FROM 8TH ROW
       GEN,801,6,1,-81,1, 91,1, 90.0*0.010*6./4. /12.
       GEN,901,6,1, 81,1,-91,1, 90.0*0.010*6./18./12.
C
```

```
HEADER NODE DATA,VERT
C
C PIPE WALL
C
       GEN,1,5,1,0.0,-1.0
C
C ENVIR
C
       -99,70.0,0.0
C
HEADER CONDUCTOR DATA,VERT
       GEN,1,5,1,1,1,99,0,  0.5*2.0*3.14*0.3/12.
       GEN,101,4,1,1,1,2,1,  90.0*3.14*.25*0.0275/12./6.
C
HEADER FLOW DATA, COOL, FID=290
C
LU DEF, TL = 80.0,     XL      = 1.0, PL      = 143.714
PA DEF, FR = 2.0,      IPDC    = 6
C
LU TANK,1000,  PL! =15.0,      TL      = -30.0
       VOL     = 1.0E-2
C
C USE NULL DEVICE: WILL SIMULATE COMPRESSOR
C
PA CONN,1000,1000,2000,
       GK      = 1.0  $ FAKE TO SATISFY INPUT REQMNTS
       DEV     = NULL
C
C DEFINE TOP END: ATTACH REFERENCE POINT
C
LU PLEN,1999
PA CONN,1999,1999,2000
       DEV     = STUBE
       DH      = 0.02/12.0
       TLEN    = 1.0
C
LU JUNC,2000,  QDOT    = 200.0
C
C NCG REFLUX TUBE: COOLED BY NATURAL CONVECTION
C
M LINE,2001,D,2001,2001,2000,        NSEG = 5
       DHS     = 0.25/12.0,   TLENT   = 2.5
       TL      = 80.0,        XL      = 0.95
       XLINC   = 0.005
       CZ      = 0.5,         CZINC   = 0.5
       TWIN    = 2011         $,PA    = STUBE
       FR      = 0.0
C
```

```
M GENT,2002,HTU,2001,2001,VERT.1,    N = 5
       UA      = 0.0
C
C VAPOR HEADER
C
PA CONN,108,2000,108
       FR      = 4.0,          DEV     = STUBE
       DH      = 0.080/12.0,   TLEN    = 0.5*2.7
LU JUNC,108
C
C RADIATOR PIPES
LU DEF, XL = 0.1,      TL = 80.0
PA DEF, FR = 0.25
C
M HX,81,C,81,81,81,PLATE.81,108,208,        NSEG    = 6
       DHS     = 0.035/12.0,   TLENT   = 3.0
       PA      = STUBE,        LU      = JUNC
       DUP     = 8.0,          TWIN    = 1081
C
C LIQUID HEADER
LU DEF, XL = 0.0,      PL = 0.0
C
LU JUNC,208
PA CONN,208,208,3000
       FR      = 4.0,          DEV     = STUBE
       DH      = 0.060/12.0,   TLEN    = 0.5*2.7
C
LU JUNC,3000
LU JUNC,4000
C
PA DEF,FR      = 4.0
       DH      = 0.060/12.0
C
C SUBCOOLER
C
M HX,91,C,96,97,96,PLATE.96,3000,4000,      NSEG    = 6
       LUINC   = -1,           PAINC   = -1,
       DHS     = 0.060/12.0,   TLENT   = 3.0
       PA      = STUBE,        LU      = JUNC
       NINC    = -1
C
PA CONN,4000,4000,5000
       DEV     = STUBE
       DH      = 0.060/12.0,   TLEN    = 3.0
C
LU TANK,5000,  XL      = 0.0
       VOL     = 1.0E-3
C
```

```
C THROTTLING VALVE
C
PA DEF, DH     = 0.012/12.0   $ VALVE THROAT
PA CONN,5000,5000,6000,
       DEV     = LOSS
       FK      = 32.0         $ BASED ON THROAT VELOCITY
C
LU DEF,XL      = 0.1, PL!     = 15.0,TL      = -30.0
LU JUNC,6000
C
C EVAPORATOR: SINGLE PASS
C
M HX,1,C,1,1,1,PLATE.1,6000,1000,   NSEG    = 4
       NINC    = 0
       DHS     = 0.125/12.0,  TLENT   = 6.0
       PA      = STUBE,       LU      = JUNC
       XL      = 0.4,         XLINC   = 0.1
       TWIN    = 1001
C
HEADER OUTPUT CALLS, COOL
       IF(LOOPCT .EQ. 0)RETURN
       CALL LMPTAB('ALL')
       CALL TIETAB('ALL')
       CALL TWNTAB('ALL')
       CALL PTHTAB('ALL')
C
```

```
HEADER OPERATIONS DATA
BUILD RAD, PLATE,VERT
BUILDF RAD, COOL
DEFMOD COOL
C
       CALL FASTIC
C
       CALL GOHOMO('COOL',0)
       LOOPCT        = 0
       CALL FASTIC
C
C CHECK CROSS CONDUCTION IN LAST PART OF CONDENSER
C
       CALL NODMAP('PLATE',91,0)
C
C CALL STDSTL TO LET REFLUX START, BUT ISOLATE FROM
C NETWORK UNTIL STEADY (SEE ALSO FLOGIC 0 LOGIC)
C
C REINITIALIZE REFLUXER (MAY HAVE CHANGED IN FASTIC)
C
       PTEST  = SNGL(PL2000)
       DO 10 ITEST = 2001,2005
              CALL GOSLIP('COOL',ITEST)
              XTEST  = 0.95 + FLOAT(ITEST-2001)*0.005
              CALL CHGLMP('COOL',ITEST,'XL',XTEST,'PL')
              CALL CHGLMP('COOL',ITEST,'PL',PTEST,'XL')
10     CONTINUE
       CALL HLDLMP('COOL',2000)
       LOOPCT        = 0
       DUPJ1999      = 0.0
       CALL STDSTL
C
C NOW RECOMBINE MODELS TO GET FINAL ANSWER
C
       LOOPCT        = 0
       CALL RELLMP('COOL',2000)
       CALL STDSTL
C
       CALL LMXTAB('COOL')
       CALL TUBTAB('COOL')
C
```

```
HEADER FLOGIC 0, COOL
C
C APPLY HEADER AC FACTORS
C
       AC108          = 1.0/(DL108*AF108*AF108)
       AC208          =-1.0/(DL208*AF208*AF208)
C
C SIMULATE COMPRESSOR: DON'T ALLOW LIQUID AT INLET
C
       IF(XL1000 .LT. 1.0)THEN
              WRITE(NOUT,10) XL1000,LOOPCT
10            FORMAT(' CHANGED QUAL FROM ',F8.6
     .               ,' AT LOOPCT = ',I4)
              CALL CHGLMP('COOL',1000,'XL',1.0,'PL')
       ENDIF
       QDOT2000       = 200.0
       CALL COMPRS('COOL',1000,0.8)
C
       IF(NSOL .LE. 0) RETURN
C
C UPDATE REFLUX HEAT TRANSFER: ASSUME CONDUCTION ACROSS FILM
C TIMES 50% (SINCE WALL ETC NEGLECTED)
C
       TTEST          = VCONDF(TL2001-ABSZRO,FI290)
       DO 30 ITEST = 2001,2005
              LTEST          = INTLMP('COOL',ITEST)
F             VTEST          = VALPHA(TL(LTEST)-ABSZRO,XL(LTEST)
     .                       ,FI290)
              VTEST          = MAX(0.1,MIN(0.9999,SQRT(VTEST)))
F             UA(INTTIE('COOL',ITEST)) =
     .                0.5*2.0*3.14*TTEST*TLEN2003/ALOG(1.0/VTEST)
30     CONTINUE
C
END OF DATA
```

Processor Output – Both Models

RELATIVE PRESSURES IN MANIFOLDS, LINES 1 THROUGH 8, RELATIVE TO 142.84 PSIA

0.8173	0.7846	0.7618	0.7473	0.7397	0.7371	0.7377	0.7378
0.0487	0.0472	0.0437	0.0386	0.0319	0.0234	0.0133	0.0000

RELATIVE PRESSURES IN MANIFOLDS, LINES 1 THROUGH 8, RELATIVE TO 142.84 PSIA

0.8174	0.7847	0.7619	0.7474	0.7398	0.7371	0.7377	0.7379
0.0487	0.0472	0.0437	0.0386	0.0319	0.0234	0.0133	0.0000

MODEL = SINDA85 DETAILED REPRESENTATION OF DIRECT COND. HEAT PUMP RADIATOR
FASTIC

FLUID SUBMODEL NAME = COOL ; FLUID NO. = 290

LOOPCT = 20
CONVERGENCE STATUS = SUBMODEL CONVERGED AS OF 16 ITERATIONS

LUMP PARAMETER TABULATION FOR SUBMODEL COOL

LUMP	TYPE	TEMP	PRESSURE	QUALITY	DENSITY	ENTHALPY	HEAT RATE	MASS RATE	ENERGY RATE
1000	JUNC	-21.22	13.19	1.000	0.1277	188.3	0.	0.	-2.8687E-03
2000	JUNC	159.9	143.7	1.000	1.070	252.5	200.0	0.	0.
101	JUNC	159.9	143.7	1.000	1.070	252.5	0.	8.9407E-08	3.8147E-05
102	JUNC	159.9	143.6	1.000	1.070	252.5	0.	-1.4901E-07	-6.1035E-05
103	JUNC	159.9	143.6	1.000	1.069	252.5	0.	2.9802E-08	0.
104	JUNC	159.9	143.6	1.000	1.069	252.5	0.	0.	-7.6294E-06
105	JUNC	159.9	143.6	1.000	1.069	252.5	0.	-5.9605E-08	-7.6294E-06
106	JUNC	159.9	143.6	1.000	1.069	252.5	0.	2.9802E-08	1.5259E-05
107	JUNC	159.9	143.6	1.000	1.069	252.5	0.	0.	7.6294E-06
108	JUNC	159.9	143.6	1.000	1.069	252.5	0.	0.	0.
11	JUNC	107.2	143.6	1.000	1.232	226.2	-10.42	0.	1.2589E-03
12	JUNC	79.89	143.5	0.9172	1.467	200.5	-10.12	0.	7.3624E-04
13	JUNC	79.80	143.3	0.7414	1.793	175.6	-9.845	0.	3.4332E-04
14	JUNC	79.72	143.2	0.5676	2.298	150.9	-9.740	0.	1.9646E-04
15	JUNC	79.65	143.0	0.3968	3.181	126.7	-9.577	0.	-9.5367E-07
16	JUNC	79.61	142.9	0.2259	5.180	102.4	-9.580	0.	-1.9073E-06
17	JUNC	79.59	142.9	6.7699E-02	12.40	79.94	-8.864	0.	0.
18	JUNC	73.30	142.9	0.	31.06	66.39	-5.351	0.	-7.2479E-05
19	JUNC	57.02	142.9	0.	31.90	56.36	-3.959	0.	-4.5824E-04
21	JUNC	106.8	143.6	1.000	1.233	226.0	-10.28	0.	2.0437E-03
22	JUNC	79.87	143.5	0.9126	1.474	199.9	-10.12	0.	2.8706E-04
23	JUNC	79.79	143.3	0.7339	1.810	174.5	-9.824	0.	1.6403E-04
24	JUNC	79.71	143.1	0.5573	2.337	149.4	-9.711	0.	9.5367E-07
25	JUNC	79.64	143.0	0.3839	3.277	124.8	-9.535	0.	-9.5367E-07
26	JUNC	79.60	142.9	0.2104	5.492	100.2	-9.539	0.	-1.9073E-06
27	JUNC	79.59	142.9	5.6199E-02	13.80	78.31	-8.478	0.	9.5367E-07
28	JUNC	71.16	142.9	0.	31.18	65.06	-5.136	0.	-1.5101E-03
29	JUNC	55.27	142.9	0.	31.99	55.30	-3.779	0.	-4.8065E-04
31	JUNC	106.6	143.6	1.000	1.234	225.9	-10.21	0.	1.8387E-03
32	JUNC	79.86	143.4	0.9095	1.479	199.4	-10.12	0.	-3.8147E-06
33	JUNC	79.78	143.3	0.7288	1.822	173.8	-9.816	0.	0.
34	JUNC	79.70	143.1	0.5503	2.364	148.4	-9.698	0.	5.7220E-06
35	JUNC	79.63	143.0	0.3751	3.345	123.6	-9.522	0.	0.
36	JUNC	79.60	142.9	0.1997	5.733	98.67	-9.533	0.	-9.5367E-07
37	JUNC	79.59	142.9	4.8344E-02	14.95	77.20	-8.221	0.	-1.3599E-03
38	JUNC	69.63	142.9	0.	31.26	64.11	-5.010	0.	-8.4305E-04
39	JUNC	53.98	142.9	0.	32.05	54.52	-3.671	0.	-2.0218E-04
41	JUNC	106.5	143.5	1.000	1.234	225.8	-10.18	0.	1.9722E-03
42	JUNC	79.85	143.4	0.9080	1.481	199.2	-10.12	0.	0.
43	JUNC	79.78	143.3	0.7263	1.827	173.4	-9.812	0.	3.8147E-06
44	JUNC	79.69	143.1	0.5470	2.377	148.0	-9.692	0.	-9.5367E-07
45	JUNC	79.63	143.0	0.3709	3.379	123.0	-9.518	0.	-9.5367E-07

LUMP PARAMETER TABULATION FOR SUBMODEL COOL

LUMP	TYPE	TEMP	PRESSURE	QUALITY	DENSITY	ENTHALPY	HEAT RATE	MASS RATE	ENERGY RATE
46	JUNC	79.60	142.9	0.1944	5.858	97.92	-9.534	0.	-1.9073E-06
47	JUNC	79.59	142.9	4.4530E-02	15.58	76.65	-8.097	0.	0.
48	JUNC	68.88	142.9	0.	31.29	63.64	-4.959	0.	-5.2137E-03
49	JUNC	53.33	142.9	0.	32.09	54.12	-3.625	0.	-7.8917E-04
51	JUNC	106.5	143.5	1.000	1.234	225.8	-10.18	0.	1.7691E-03
52	JUNC	79.85	143.4	0.9078	1.481	199.2	-10.13	0.	2.6512E-04
53	JUNC	79.77	143.3	0.7261	1.828	173.4	-9.812	0.	0.
54	JUNC	79.69	143.1	0.5467	2.378	147.9	-9.692	0.	1.9073E-06
55	JUNC	79.63	143.0	0.3705	3.382	122.9	-9.519	0.	1.9073E-06
56	JUNC	79.59	142.9	0.1938	5.874	97.83	-9.543	0.	0.
57	JUNC	79.58	142.9	4.3897E-02	15.69	76.56	-8.095	0.	-2.9049E-03
58	JUNC	68.71	142.9	0.	31.30	63.54	-4.957	0.	-1.7095E-03
59	JUNC	53.16	142.9	0.	32.09	54.02	-3.622	0.	-4.3297E-04
61	JUNC	106.5	143.5	1.000	1.234	225.8	-10.21	0.	1.9693E-03
62	JUNC	79.85	143.4	0.9085	1.480	199.3	-10.13	0.	9.5367E-07
63	JUNC	79.77	143.3	0.7273	1.825	173.6	-9.816	0.	9.5367E-07
64	JUNC	79.69	143.1	0.5484	2.371	148.2	-9.699	0.	-6.0558E-04
65	JUNC	79.62	143.0	0.3727	3.364	123.2	-9.532	0.	0.
66	JUNC	79.59	142.9	0.1959	5.823	98.12	-9.584	0.	1.9073E-06
67	JUNC	79.58	142.9	4.4606E-02	15.56	76.66	-8.197	0.	1.9073E-06
68	JUNC	68.78	142.9	0.	31.30	63.58	-4.996	0.	2.3603E-04
69	JUNC	53.17	142.9	0.	32.09	54.03	-3.646	0.	1.2064E-04
71	JUNC	106.3	143.5	1.000	1.235	225.7	-10.35	0.	1.1854E-03
72	JUNC	79.85	143.4	0.9080	1.481	199.2	-10.23	0.	9.5367E-07
73	JUNC	79.77	143.3	0.7273	1.825	173.6	-9.898	0.	3.7479E-04
74	JUNC	79.68	143.1	0.5485	2.371	148.2	-9.803	0.	0.
75	JUNC	79.62	143.0	0.3719	3.370	123.1	-9.681	0.	-9.5367E-07
76	JUNC	79.59	142.9	0.1913	5.935	97.47	-9.895	0.	-9.5367E-07
77	JUNC	79.58	142.9	3.9372E-02	16.52	75.91	-8.322	0.	0.
78	JUNC	67.79	142.9	0.	31.35	62.97	-4.997	0.	2.2411E-04
79	JUNC	52.49	142.9	0.	32.13	53.62	-3.611	0.	3.6430E-04
81	JUNC	104.2	143.5	1.000	1.243	224.7	-11.73	0.	1.3647E-03
82	JUNC	79.83	143.4	0.8708	1.540	193.9	-12.94	0.	5.4550E-04
83	JUNC	79.74	143.2	0.6650	1.984	164.7	-12.30	0.	2.8610E-04
84	JUNC	79.65	143.0	0.4652	2.756	136.4	-11.95	0.	9.5367E-07
85	JUNC	79.60	142.9	0.2714	4.436	108.8	-11.58	0.	-9.5367E-07
86	JUNC	79.57	142.9	9.4758E-02	10.01	83.77	-10.56	0.	-9.5367E-07
87	JUNC	78.69	142.9	0.	30.78	69.77	-5.895	0.	8.8549E-04
88	JUNC	61.67	142.9	0.	31.67	59.20	-4.448	0.	1.1415E-03
89	JUNC	49.42	142.8	0.	32.28	51.76	-3.131	0.	7.1907E-04
201	JUNC	57.02	142.9	0.	31.90	56.36	0.	0.	0.
202	JUNC	56.16	142.9	0.	31.94	55.84	0.	0.	0.
203	JUNC	55.44	142.9	0.	31.98	55.41	0.	0.	0.
204	JUNC	54.92	142.9	0.	32.01	55.09	0.	0.	0.
205	JUNC	54.57	142.9	0.	32.02	54.88	0.	1.1921E-07	7.6294E-06
206	JUNC	54.34	142.9	0.	32.04	54.74	0.	0.	0.
207	JUNC	54.08	142.8	0.	32.05	54.58	0.	0.	0.
208	JUNC	53.45	142.8	0.	32.08	54.20	0.	0.	0.
3000	JUNC	53.45	142.8	0.	32.08	54.20	0.	0.	8.6975E-04

LUMP PARAMETER TABULATION FOR SUBMODEL COOL

LUMP	TYPE	TEMP	PRESSURE	QUALITY	DENSITY	ENTHALPY	HEAT RATE	MASS RATE	ENERGY RATE
4000	JUNC	39.82	142.7	0.	32.74	46.03	0.	0.	-2.1362E-04
99	JUNC	50.55	142.8	0.	32.22	52.45	-5.446	0.	6.6757E-06
98	JUNC	48.10	142.8	0.	32.34	50.97	-4.599	0.	6.1989E-06
97	JUNC	46.10	142.8	0.	32.44	49.77	-3.734	0.	7.1573E-04
96	JUNC	44.63	142.8	0.	32.51	48.89	-2.745	0.	3.6573E-04
95	JUNC	43.43	142.7	0.	32.57	48.17	-2.236	0.	-2.5415E-04
94	JUNC	42.38	142.7	0.	32.62	47.55	-1.945	0.	-3.4571E-04
93	JUNC	41.45	142.7	0.	32.67	47.00	-1.721	0.	-2.3003E-03
92	JUNC	40.63	142.7	0.	32.70	46.51	-1.532	0.	-1.0424E-03
91	JUNC	39.82	142.7	0.	32.74	46.03	-1.492	0.	3.3379E-06
5000	JUNC	39.82	142.5	0.	32.74	46.03	0.	0.	8.0872E-04
6000	JUNC	-44.24	14.51	0.2654	0.5587	46.02	0.	0.	1.9012E-02
1	JUNC	-44.71	14.34	0.8730	0.1698	156.5	344.3	0.	3.0518E-05
2	JUNC	-45.36	14.11	0.9885	0.1478	177.4	65.04	0.	-1.5259E-05
3	JUNC	-33.61	13.91	1.000	0.1393	183.7	19.69	0.	0.
4	JUNC	-25.93	13.71	1.000	0.1345	186.5	8.718	0.	-7.5150E-03
5	JUNC	-22.60	13.50	1.000	0.1313	187.8	3.856	0.	-3.3774E-03
6	JUNC	-21.17	13.30	1.000	0.1287	188.3	1.733	0.	-3.7746E-03
1999	PLEN	80.00	143.7	1.000	1.353	212.3	0.	0.	0.

MODEL = SINDA85
FASTIC

DETAILED REPRESENTATION OF DIRECT COND. HEAT PUMP RADIATOR

FLUID SUBMODEL NAME = COOL ; FLUID NO. = 290

LOOPCT = 20
CONVERGENCE STATUS = SUBMODEL CONVERGED AS OF 16 ITERATIONS

TIE PARAMETER TABULATION FOR SUBMODEL COOL

TIE	TYPE	UA	QTIE	LUMP	TEMP.	NODE	TEMP.	PATH 1	FRACT	PATH 2	FRACT
11	HTNC	0.3645	-10.42	11	107.2	PLATE.11	78.61	11	1.0000	12	0.5000
12	HTNC	4.081	-10.12	12	79.89	PLATE.12	77.41	12	0.5000	13	0.5000
13	HTNC	2.583	-9.845	13	79.80	PLATE.13	75.99	13	0.5000	14	0.5000
14	HTNC	2.062	-9.740	14	79.72	PLATE.14	74.99	14	0.5000	15	0.5000
15	HTNC	1.613	-9.577	15	79.65	PLATE.15	73.71	15	0.5000	16	0.5000
16	HTNC	1.204	-9.580	16	79.61	PLATE.16	71.65	16	0.5000	17	0.5000
17	HTNC	0.5923	-8.864	17	79.59	PLATE.17	64.63	17	0.5000	18	0.5000
18	HTNC	0.2170	-5.351	18	73.30	PLATE.18	48.65	18	0.5000	19	0.5000
19	HTNC	0.2266	-3.959	19	57.02	PLATE.19	39.55	19	0.5000	20	1.0000
21	HTNC	0.3588	-10.28	21	106.8	PLATE.21	78.19	21	1.0000	22	0.5000
22	HTNC	3.929	-10.12	22	79.87	PLATE.22	77.30	22	0.5000	23	0.5000
23	HTNC	2.522	-9.824	23	79.79	PLATE.23	75.90	23	0.5000	24	0.5000
24	HTNC	2.006	-9.711	24	79.71	PLATE.24	74.87	24	0.5000	25	0.5000
25	HTNC	1.556	-9.535	25	79.64	PLATE.25	73.51	25	0.5000	26	0.5000
26	HTNC	1.145	-9.539	26	79.60	PLATE.26	71.27	26	0.5000	27	0.5000
27	HTNC	0.5223	-8.478	27	79.59	PLATE.27	63.36	27	0.5000	28	0.5000
28	HTNC	0.2183	-5.136	28	71.16	PLATE.28	47.64	28	0.5000	29	0.5000
29	HTNC	0.2277	-3.779	29	55.27	PLATE.29	38.67	29	0.5000	30	1.0000
31	HTNC	0.3553	-10.21	31	106.6	PLATE.31	77.87	31	1.0000	32	0.5000
32	HTNC	3.834	-10.12	32	79.86	PLATE.32	77.22	32	0.5000	33	0.5000
33	HTNC	2.484	-9.816	33	79.78	PLATE.33	75.83	33	0.5000	34	0.5000
34	HTNC	1.971	-9.698	34	79.70	PLATE.34	74.78	34	0.5000	35	0.5000
35	HTNC	1.522	-9.522	35	79.63	PLATE.35	73.38	35	0.5000	36	0.5000
36	HTNC	1.105	-9.533	36	79.60	PLATE.36	70.97	36	0.5000	37	0.5000
37	HTNC	0.4752	-8.221	37	79.59	PLATE.37	62.29	37	0.5000	38	0.5000
38	HTNC	0.2192	-5.010	38	69.63	PLATE.38	46.78	38	0.5000	39	0.5000
39	HTNC	0.2284	-3.671	39	53.98	PLATE.39	37.91	39	0.5000	40	1.0000
41	HTNC	0.3537	-10.18	41	106.5	PLATE.41	77.70	41	1.0000	42	0.5000
42	HTNC	3.791	-10.12	42	79.85	PLATE.42	77.18	42	0.5000	43	0.5000
43	HTNC	2.466	-9.812	43	79.78	PLATE.43	75.80	43	0.5000	44	0.5000
44	HTNC	1.954	-9.692	44	79.69	PLATE.44	74.74	44	0.5000	45	0.5000
45	HTNC	1.506	-9.518	45	79.63	PLATE.45	73.31	45	0.5000	46	0.5000
46	HTNC	1.085	-9.534	46	79.60	PLATE.46	70.81	46	0.5000	47	0.5000
47	HTNC	0.4527	-8.097	47	79.59	PLATE.47	61.70	47	0.5000	48	0.5000
48	HTNC	0.2196	-4.959	48	68.88	PLATE.48	46.31	48	0.5000	49	0.5000
49	HTNC	0.2288	-3.625	49	53.33	PLATE.49	37.49	49	0.5000	50	1.0000
51	HTNC	0.3536	-10.18	51	106.5	PLATE.51	77.67	51	1.0000	52	0.5000
52	HTNC	3.787	-10.13	52	79.85	PLATE.52	77.18	52	0.5000	53	0.5000
53	HTNC	2.465	-9.812	53	79.77	PLATE.53	75.79	53	0.5000	54	0.5000
54	HTNC	1.953	-9.692	54	79.69	PLATE.54	74.73	54	0.5000	55	0.5000
55	HTNC	1.505	-9.519	55	79.63	PLATE.55	73.30	55	0.5000	56	0.5000
56	HTNC	1.083	-9.543	56	79.59	PLATE.56	70.79	56	0.5000	57	0.5000

MODEL = SINDA85
FASTIC

DETAILED REPRESENTATION OF DIRECT COND. HEAT PUMP RADIATOR

TIE PARAMETER TABULATION FOR SUBMODEL COOL

TIE	TYPE	UA	QTIE	LUMP	TEMP.	NODE	TEMP.	PATH 1	FRACT	PATH 2	FRACT
57	HTNC	0.4491	-8.095	57	79.58	PLATE.57	61.56	57	0.5000	58	0.5000
58	HTNC	0.2197	-4.957	58	68.71	PLATE.58	46.16	58	0.5000	59	0.5000
59	HTNC	0.2289	-3.622	59	53.16	PLATE.59	37.34	59	0.5000	60	1.0000
61	HTNC	0.3546	-10.21	61	106.5	PLATE.61	77.69	61	1.0000	62	0.5000
62	HTNC	3.808	-10.13	62	79.85	PLATE.62	77.19	62	0.5000	63	0.5000
63	HTNC	2.475	-9.816	63	79.77	PLATE.63	75.80	63	0.5000	64	0.5000
64	HTNC	1.963	-9.699	64	79.69	PLATE.64	74.75	64	0.5000	65	0.5000
65	HTNC	1.514	-9.532	65	79.62	PLATE.65	73.33	65	0.5000	66	0.5000
66	HTNC	1.092	-9.584	66	79.59	PLATE.66	70.81	66	0.5000	67	0.5000
67	HTNC	0.4535	-8.197	67	79.58	PLATE.67	61.51	67	0.5000	68	0.5000
68	HTNC	0.2197	-4.996	68	68.78	PLATE.68	46.04	68	0.5000	69	0.5000
69	HTNC	0.2289	-3.646	69	53.17	PLATE.69	37.24	69	0.5000	70	1.0000
71	HTNC	0.3576	-10.35	71	106.3	PLATE.71	77.33	71	1.0000	72	0.5000
72	HTNC	3.813	-10.23	72	79.85	PLATE.72	77.16	72	0.5000	73	0.5000
73	HTNC	2.494	-9.898	73	79.77	PLATE.73	75.80	73	0.5000	74	0.5000
74	HTNC	1.978	-9.803	74	79.68	PLATE.74	74.73	74	0.5000	75	0.5000
75	HTNC	1.524	-9.681	75	79.62	PLATE.75	73.27	75	0.5000	76	0.5000
76	HTNC	1.088	-9.895	76	79.59	PLATE.76	70.49	76	0.5000	77	0.5000
77	HTNC	0.4266	-8.322	77	79.58	PLATE.77	60.07	77	0.5000	78	0.5000
78	HTNC	0.2203	-4.997	78	67.79	PLATE.78	45.11	78	0.5000	79	0.5000
79	HTNC	0.2293	-3.611	79	52.49	PLATE.79	36.74	79	0.5000	80	1.0000
81	HTNC	0.3824	-11.73	81	104.2	PLATE.81	73.55	81	1.0000	82	0.5000
82	HTNC	3.430	-12.94	82	79.83	PLATE.82	76.06	82	0.5000	83	0.5000
83	HTNC	2.442	-12.30	83	79.74	PLATE.83	74.71	83	0.5000	84	0.5000
84	HTNC	1.879	-11.95	84	79.65	PLATE.84	73.30	84	0.5000	85	0.5000
85	HTNC	1.386	-11.58	85	79.60	PLATE.85	71.24	85	0.5000	86	0.5000
86	HTNC	0.7682	-10.56	86	79.57	PLATE.86	65.83	86	0.5000	87	0.5000
87	HTNC	0.2139	-5.895	87	78.69	PLATE.87	51.12	87	0.5000	88	0.5000
88	HTNC	0.2239	-4.448	88	61.67	PLATE.88	41.80	88	0.5000	89	0.5000
89	HTNC	0.2311	-3.131	89	49.42	PLATE.89	35.87	89	0.5000	90	1.0000
99	HTNC	0.6346	-5.446	99	50.55	PLATE.99	41.97	100	1.0000	99	0.5000
100	HTNC	0.6210	-4.599	98	48.10	PLATE.98	40.69	99	0.5000	98	0.5000
101	HTNC	0.6098	-3.734	97	46.10	PLATE.97	39.98	98	0.5000	97	0.5000
102	HTNC	0.6016	-2.745	96	44.63	PLATE.96	40.07	97	0.5000	96	0.5000
103	HTNC	0.5949	-2.236	95	43.43	PLATE.95	39.67	96	0.5000	95	0.5000
104	HTNC	0.5891	-1.945	94	42.38	PLATE.94	39.08	95	0.5000	94	0.5000
105	HTNC	0.5839	-1.721	93	41.45	PLATE.93	38.51	94	0.5000	93	0.5000
106	HTNC	0.5793	-1.532	92	40.63	PLATE.92	37.98	93	0.5000	92	0.5000
107	HTNC	0.5748	-1.492	91	39.82	PLATE.91	37.23	92	0.5000	91	1.0000
1	HTNC	13.93	344.3	1	-44.71	PLATE.1	-20.00	1	1.0000	2	0.5000
2	HTNC	2.564	65.04	2	-45.36	PLATE.1	-20.00	2	0.5000	3	0.5000
3	HTNC	1.446	19.69	3	-33.61	PLATE.1	-20.00	3	0.5000	4	0.5000
4	HTNC	1.472	8.718	4	-25.93	PLATE.1	-20.00	4	0.5000	5	0.5000
5	HTNC	1.483	3.856	5	-22.60	PLATE.1	-20.00	5	0.5000	6	0.5000
6	HTNC	1.487	1.733	6	-21.17	PLATE.1	-20.00	6	0.5000	7	1.0000

MODEL = SINDA85
FASTIC

DETAILED REPRESENTATION OF DIRECT COND. HEAT PUMP RADIATOR

FLUID SUBMODEL NAME = COOL ; FLUID NO. = 290

LOOPCT = 20
CONVERGENCE STATUS = SUBMODEL CONVERGED AS OF 16 ITERATIONS

PATH PARAMETER TABULATION FOR SUBMODEL COOL

PATH	TYPE	LMP 1	LMP 2	DUP I	DUP J	STAT	XL UPSTRM	FLOWRATE	DELTA PRES	FLOW REGIMES OR REYNOLDS NO.	
1000	NULL	1000	2000	1.0	1.0	NORM	1.000	3.115	-130.5		
1999	STUBE	1999	2000	1.0	1.0	NORM	1.000	0.	0.	0.	0.
101	STUBE	2000	101	1.0	1.0	NORM	1.000	3.115	6.0761E-02	26078.	26078.
102	STUBE	101	102	1.0	1.0	NORM	1.000	2.720	3.2687E-02	22773.	22773.
103	STUBE	102	103	1.0	1.0	NORM	1.000	2.333	2.2803E-02	19530.	19530.
104	STUBE	103	104	1.0	1.0	NORM	1.000	1.950	1.4447E-02	16326.	16326.
105	STUBE	104	105	1.0	1.0	NORM	1.000	1.570	7.6820E-03	13139.	13139.
106	STUBE	105	106	1.0	1.0	NORM	1.000	1.189	2.6157E-03	9953.8	9953.8
107	STUBE	106	107	1.0	1.0	NORM	1.000	0.8071	-6.1105E-04	6756.7	6756.7
108	STUBE	107	108	1.0	1.0	NORM	1.000	0.4211	-1.3489E-04	3525.1	3525.1
11	STUBE	101	11	1.0	1.0	NORM	1.000	0.3948	4.0967E-02	7554.8	8225.1
12	STUBE	11	12	1.0	1.0	NORM	1.000	0.3948	0.1168	8225.1	ANNULAR
13	STUBE	12	13	1.0	1.0	NORM	0.9172	0.3948	0.1653	ANNULAR	ANNULAR
14	STUBE	13	14	1.0	1.0	NORM	0.7414	0.3948	0.1717	ANNULAR	ANNULAR
15	STUBE	14	15	1.0	1.0	NORM	0.5676	0.3948	0.1432	ANNULAR	ANNULAR
16	STUBE	15	16	1.0	1.0	NORM	0.3968	0.3948	8.0456E-02	ANNULAR	ANNULAR
17	STUBE	16	17	1.0	1.0	NORM	0.2259	0.3948	2.4369E-02	ANNULAR	SLUG
18	STUBE	17	18	1.0	1.0	NORM	6.7699E-02	0.3948	9.4398E-03	SLUG	655.70
19	STUBE	18	19	1.0	1.0	NORM	0.	0.3948	1.0767E-02	655.70	595.87
20	STUBE	19	201	1.0	1.0	NORM	0.	0.3948	5.5942E-03	595.87	595.87
21	STUBE	102	21	1.0	1.0	NORM	1.000	0.3874	3.9619E-02	7411.8	8074.0
22	STUBE	21	22	1.0	1.0	NORM	1.000	0.3874	0.1140	8074.0	ANNULAR
23	STUBE	22	23	1.0	1.0	NORM	0.9126	0.3874	0.1614	ANNULAR	ANNULAR
24	STUBE	23	24	1.0	1.0	NORM	0.7339	0.3874	0.1665	ANNULAR	ANNULAR
25	STUBE	24	25	1.0	1.0	NORM	0.5573	0.3874	0.1350	ANNULAR	ANNULAR
26	STUBE	25	26	1.0	1.0	NORM	0.3839	0.3874	7.2994E-02	ANNULAR	ANNULAR
27	STUBE	26	27	1.0	1.0	NORM	0.2104	0.3874	2.2197E-02	ANNULAR	SLUG
28	STUBE	27	28	1.0	1.0	NORM	5.6199E-02	0.3874	9.6148E-03	SLUG	635.20
29	STUBE	28	29	1.0	1.0	NORM	0.	0.3874	1.0653E-02	635.20	578.65
30	STUBE	29	202	1.0	1.0	NORM	0.	0.3874	5.5193E-03	578.65	581.64
31	STUBE	103	31	1.0	1.0	NORM	1.000	0.3828	3.8801E-02	7324.6	7982.3
32	STUBE	31	32	1.0	1.0	NORM	1.000	0.3828	0.1123	7982.3	ANNULAR
33	STUBE	32	33	1.0	1.0	NORM	0.9095	0.3828	0.1591	ANNULAR	ANNULAR
34	STUBE	33	34	1.0	1.0	NORM	0.7288	0.3828	0.1632	ANNULAR	ANNULAR
35	STUBE	34	35	1.0	1.0	NORM	0.5503	0.3828	0.1299	ANNULAR	ANNULAR
36	STUBE	35	36	1.0	1.0	NORM	0.3751	0.3828	6.8217E-02	ANNULAR	ANNULAR
37	STUBE	36	37	1.0	1.0	NORM	0.1997	0.3828	2.0804E-02	ANNULAR	SLUG
38	STUBE	37	38	1.0	1.0	NORM	4.8344E-02	0.3828	9.7101E-03	SLUG	622.09
39	STUBE	38	39	1.0	1.0	NORM	0.	0.3828	1.0593E-02	622.09	567.56
40	STUBE	39	203	1.0	1.0	NORM	0.	0.3828	5.4780E-03	567.56	572.41
41	STUBE	104	41	1.0	1.0	NORM	1.000	0.3807	3.8424E-02	7284.0	7939.7
42	STUBE	41	42	1.0	1.0	NORM	1.000	0.3807	0.1115	7939.7	ANNULAR

MODEL = SINDA85
FASTIC

DETAILED REPRESENTATION OF DIRECT COND. HEAT PUMP RADIATOR

PATH PARAMETER TABULATION FOR SUBMODEL COOL

PATH	TYPE	LMP 1	LMP 2	DUP I	DUP J	STAT	XL UPSTRM	FLOWRATE	DELTA PRES	FLOW REGIMES OR REYNOLDS NO.	
43	STUBE	42	43	1.0	1.0	NORM	0.9080	0.3807	0.1579	ANNULAR	ANNULAR
44	STUBE	43	44	1.0	1.0	NORM	0.7263	0.3807	0.1617	ANNULAR	ANNULAR
45	STUBE	44	45	1.0	1.0	NORM	0.5470	0.3807	0.1274	ANNULAR	ANNULAR
46	STUBE	45	46	1.0	1.0	NORM	0.3709	0.3807	6.5886E-02	ANNULAR	ANNULAR
47	STUBE	46	47	1.0	1.0	NORM	0.1944	0.3807	2.0143E-02	ANNULAR	SLUG
48	STUBE	47	48	1.0	1.0	NORM	4.4530E-02	0.3807	9.7484E-03	SLUG	615.89
49	STUBE	48	49	1.0	1.0	NORM	0.	0.3807	1.0567E-02	615.89	562.26
50	STUBE	49	204	1.0	1.0	NORM	0.	0.3807	5.4616E-03	562.26	567.51
51	STUBE	105	51	1.0	1.0	NORM	1.000	0.3805	3.8396E-02	7280.9	7936.6
52	STUBE	51	52	1.0	1.0	NORM	1.000	0.3805	0.1114	7936.6	ANNULAR
53	STUBE	52	53	1.0	1.0	NORM	0.9078	0.3805	0.1578	ANNULAR	ANNULAR
54	STUBE	53	54	1.0	1.0	NORM	0.7261	0.3805	0.1615	ANNULAR	ANNULAR
55	STUBE	54	55	1.0	1.0	NORM	0.5467	0.3805	0.1271	ANNULAR	ANNULAR
56	STUBE	55	56	1.0	1.0	NORM	0.3705	0.3805	6.5650E-02	ANNULAR	ANNULAR
57	STUBE	56	57	1.0	1.0	NORM	0.1938	0.3805	2.0073E-02	ANNULAR	SLUG
58	STUBE	57	58	1.0	1.0	NORM	4.3897E-02	0.3805	9.7553E-03	SLUG	615.03
59	STUBE	58	59	1.0	1.0	NORM	0.	0.3805	1.0569E-02	615.03	561.48
60	STUBE	59	205	1.0	1.0	NORM	0.	0.3805	5.4651E-03	561.48	566.12
61	STUBE	106	61	1.0	1.0	NORM	1.000	0.3819	3.8635E-02	7307.6	7965.5
62	STUBE	61	62	1.0	1.0	NORM	1.000	0.3819	0.1120	7965.5	ANNULAR
63	STUBE	62	63	1.0	1.0	NORM	0.9085	0.3819	0.1587	ANNULAR	ANNULAR
64	STUBE	63	64	1.0	1.0	NORM	0.7273	0.3819	0.1626	ANNULAR	ANNULAR
65	STUBE	64	65	1.0	1.0	NORM	0.5484	0.3819	0.1286	ANNULAR	ANNULAR
66	STUBE	65	66	1.0	1.0	NORM	0.3727	0.3819	6.6917E-02	ANNULAR	ANNULAR
67	STUBE	66	67	1.0	1.0	NORM	0.1959	0.3819	2.0310E-02	ANNULAR	SLUG
68	STUBE	67	68	1.0	1.0	NORM	4.4606E-02	0.3819	9.7755E-03	SLUG	617.52
69	STUBE	68	69	1.0	1.0	NORM	0.	0.3819	1.0606E-02	617.52	563.59
70	STUBE	69	206	1.0	1.0	NORM	0.	0.3819	5.4878E-03	563.59	567.43
71	STUBE	107	71	1.0	1.0	NORM	1.000	0.3861	3.9292E-02	7386.7	8054.2
72	STUBE	71	72	1.0	1.0	NORM	1.000	0.3861	0.1139	8054.2	ANNULAR
73	STUBE	72	73	1.0	1.0	NORM	0.9080	0.3861	0.1613	ANNULAR	ANNULAR
74	STUBE	73	74	1.0	1.0	NORM	0.7273	0.3861	0.1650	ANNULAR	ANNULAR
75	STUBE	74	75	1.0	1.0	NORM	0.5485	0.3861	0.1309	ANNULAR	ANNULAR
76	STUBE	75	76	1.0	1.0	NORM	0.3719	0.3861	6.7703E-02	ANNULAR	ANNULAR
77	STUBE	76	77	1.0	1.0	NORM	0.1913	0.3861	2.0026E-02	ANNULAR	SLUG
78	STUBE	77	78	1.0	1.0	NORM	3.9372E-02	0.3861	9.9685E-03	SLUG	620.60
79	STUBE	78	79	1.0	1.0	NORM	0.	0.3861	1.0760E-02	620.60	567.42
80	STUBE	79	207	1.0	1.0	NORM	0.	0.3861	5.5584E-03	567.42	572.69
81	STUBE	108	81	1.0	1.0	NORM	1.000	0.4211	4.4886E-02	8057.3	8816.3
82	STUBE	81	82	1.0	1.0	NORM	1.000	0.4211	0.1337	8816.3	ANNULAR
83	STUBE	82	83	1.0	1.0	NORM	0.8708	0.4211	0.1853	ANNULAR	ANNULAR
84	STUBE	83	84	1.0	1.0	NORM	0.6650	0.4211	0.1765	ANNULAR	ANNULAR
85	STUBE	84	85	1.0	1.0	NORM	0.4652	0.4211	0.1140	ANNULAR	ANNULAR
86	STUBE	85	86	1.0	1.0	NORM	0.2714	0.4211	4.5271E-02	ANNULAR	ANNULAR
87	STUBE	86	87	1.0	1.0	NORM	9.4758E-02	0.4211	8.8929E-03	SLUG	721.95
88	STUBE	87	88	1.0	1.0	NORM	0.	0.4211	1.1234E-02	721.95	653.01
89	STUBE	88	89	1.0	1.0	NORM	0.	0.4211	1.1974E-02	653.01	608.02
90	STUBE	89	208	1.0	1.0	NORM	0.	0.4211	6.1112E-03	608.02	622.40
201	STUBE	201	202	1.0	1.0	NORM	0.	0.3948	1.5715E-03	347.59	345.84

MODEL = SINDA85
FASTIC

DETAILED REPRESENTATION OF DIRECT COND. HEAT PUMP RADIATOR

PATH PARAMETER TABULATION FOR SUBMODEL COOL

PATH	TYPE	LMP 1	LMP 2	DUP I	DUP J	STAT	XL UPSTRM	FLOWRATE	DELTA PRES	FLOW REGIMES OR REYNOLDS NO.	
202	STUBE	202	203	1.0	1.0	NORM	0.	0.7822	3.4246E-03	685.13	682.29
203	STUBE	203	204	1.0	1.0	NORM	0.	1.165	5.1001E-03	1016.2	1013.1
204	STUBE	204	205	1.0	1.0	NORM	0.	1.546	6.7712E-03	1344.2	1341.4
205	STUBE	205	206	1.0	1.0	NORM	0.	1.926	8.4498E-03	1671.7	1669.4
206	STUBE	206	207	1.0	1.0	NORM	0.	2.308	1.0155E-02	2000.4	1997.3
207	STUBE	207	208	1.0	1.0	NORM	0.	2.694	1.3254E-02	2331.4	2322.9
208	STUBE	208	3000	1.0	1.0	NORM	0.	3.115	1.9053E-02	2686.0	2686.0
100	STUBE	3000	99	1.0	1.0	NORM	0.	3.115	8.3837E-03	2686.0	2641.2
99	STUBE	99	98	1.0	1.0	NORM	0.	3.115	1.6385E-02	2641.2	2604.0
98	STUBE	98	97	1.0	1.0	NORM	0.	3.115	1.6014E-02	2604.0	2574.2
97	STUBE	97	96	1.0	1.0	NORM	0.	3.115	1.5722E-02	2574.2	2552.5
96	STUBE	96	95	1.0	1.0	NORM	0.	3.115	1.5495E-02	2552.5	2534.9
95	STUBE	95	94	1.0	1.0	NORM	0.	3.115	1.5304E-02	2534.9	2519.8
94	STUBE	94	93	1.0	1.0	NORM	0.	3.115	1.5136E-02	2519.8	2506.5
93	STUBE	93	92	1.0	1.0	NORM	0.	3.115	1.4988E-02	2506.5	2494.7
92	STUBE	92	91	1.0	1.0	NORM	0.	3.115	1.4851E-02	2494.7	2483.2
91	STUBE	91	4000	1.0	1.0	NORM	0.	3.115	7.3989E-03	2483.2	2483.2
4000	STUBE	4000	5000	1.0	1.0	NORM	0.	3.115	0.1332	2483.2	2483.2
5000	LOSS	5000	6000	1.0	1.0	NORM	0.	3.115	128.0		
1	STUBE	6000	1	1.0	1.0	NORM	0.2654	3.115	0.1669	ANNULAR	ANNULAR
2	STUBE	1	2	1.0	1.0	NORM	0.8730	3.115	0.2293	ANNULAR	ANNULAR
3	STUBE	2	3	1.0	1.0	NORM	0.9885	3.115	0.2049	ANNULAR	23972.
4	STUBE	3	4	1.0	1.0	NORM	1.000	3.115	0.1989	23972.	23567.
5	STUBE	4	5	1.0	1.0	NORM	1.000	3.115	0.2036	23567.	23395.
6	STUBE	5	6	1.0	1.0	NORM	1.000	3.115	0.2076	23395.	23322.
7	STUBE	6	1000	1.0	1.0	NORM	1.000	3.115	0.1036	23322.	23324.

MODEL = SINDA85
NODMAP

DETAILED REPRESENTATION OF DIRECT COND. HEAT PUMP RADIATOR

NODE PLATE 91

A QMAP OF INPUT ARITH NODE PLATE 91 (INTERNAL 73)

THE PARAMETERS OF NODE PLATE 91 ARE:

TEMPERATURE	=	37.2268	(DEG.)
CAPACITANCE	=	0.	(ENERGY/DEG)
NET SOURCE/SINK	=	1.49218	(ENERGY/TIME, INCLUDES TIES)
CAP./SUM OF COND.	=	0.	(TIME, INCLUDES TIES)

THE ADJOINING NODES TO NODE PLATE 91 ARE:

NODE INPUT	(INTERNAL)	CONDUCTOR INPUT	(INTERNAL)	TYPE	CONDUCTOR VALUE	% OF TYPE	% OF TOTAL	HEAT TRANSFER RATE (ENERGY/TIME)	TEMPERATURE OF ADJOINING NODE
PLATE	92(74)	91	(65)	LINEAR	7.500000E-02	50.0	26.3	5.674896E-02	37.9835
PLATE	81(64)	901	(136)	LINEAR	7.500000E-02	50.0	26.3	2.72449	73.5534
PLATE	999(82)	991	(217)	RADIAT	0.152889	100.0	43.4	-4.27329	0.

THE ADJOINING LUMPS TO NODE PLATE 91 ARE:

LUMP MODEL.INPUT	(INTL)	TIE INPUT	(INTL)	TYPE	DUPN	TIE CONDUCTANCE	HEAT RATE	ADJOINING LUMP TEMPERATURE	PROPERTIES DUPL
COOL.91	(101)	107	(81)	HTNC	1.000	0.574797	1.49218	39.8226	1.000

THE TOTALS ON NODE PLATE 91 ARE:

LINEAR HEAT TRANSFER (CONDUCTION/CONVECTION)	2.78124	
RADIATION HEAT TRANSFER	-4.27329	
TIE HEAT TRANSFER (FROM LUMPS)	1.49218	
HEAT SOURCE/SINKS APPLIED	0.	
	1.337528E-04	(ENERGY/TIME)
EFFECTIVE ERN TEMPERATURE	0.	

MODEL = SINDA85
FASTIC

SYSTEM REPRESENTATION OF DIRECT COND. HEAT PUMP RADIATOR

FLUID SUBMODEL NAME = COOL ; FLUID NO. = 290

LOOPCT = 133
CONVERGENCE STATUS = SUBMODEL CONVERGED AS OF 132 ITERATIONS. ENERGY STABLE BUT UNBALANCED

LUMP PARAMETER TABULATION FOR SUBMODEL COOL

LUMP	TYPE	TEMP	PRESSURE	QUALITY	DENSITY	ENTHALPY	HEAT RATE	MASS RATE	ENERGY RATE
1000	TANK	-22.80	13.08	1.000	0.1270	187.8	0.	0.	-1.3428E-03
2001	TANK	79.99	143.7	0.9550	1.414	205.9	0.	0.	0.
2002	TANK	79.99	143.7	0.9550	1.414	205.9	0.	0.	0.
2003	TANK	79.99	143.7	0.9600	1.407	206.6	0.	0.	0.
2004	TANK	79.98	143.7	0.9650	1.400	207.3	0.	0.	0.
2005	TANK	79.98	143.7	0.9650	1.400	207.3	0.	0.	0.
5000	TANK	40.76	142.4	0.	32.70	46.58	0.	0.	5.6000E-03
2000	JUNC	158.9	143.7	1.000	1.073	252.0	200.0	0.	0.
108	JUNC	158.9	143.5	1.000	1.071	252.0	0.	0.	0.
81	JUNC	79.86	143.4	0.6319	2.084	160.1	-16.58	0.	1.1444E-05
82	JUNC	79.77	143.3	0.2308	5.100	103.2	-15.54	0.	1.5583E-03
83	JUNC	79.65	143.0	0.1312	7.956	88.98	-15.17	0.	4.0340E-04
84	JUNC	79.55	142.8	0.1240	8.263	87.90	-14.77	3.7253E-08	2.6608E-04
85	JUNC	78.26	142.8	0.	30.80	69.50	-8.962	-2.9802E-08	1.9827E-03
86	JUNC	54.37	142.8	0.	32.03	54.76	-5.739	0.	3.0327E-04
208	JUNC	54.37	142.8	0.	32.03	54.75	0.	0.	3.8300E-03
3000	JUNC	54.37	142.7	0.	32.03	54.75	0.	0.	0.
4000	JUNC	40.76	142.6	0.	32.70	46.58	0.	0.	-3.5400E-03
96	JUNC	50.19	142.7	0.	32.24	52.23	-7.867	0.	-4.6682E-04
95	JUNC	47.03	142.7	0.	32.40	50.33	-5.909	0.	-6.8283E-04
94	JUNC	44.95	142.6	0.	32.50	49.09	-3.874	0.	-6.2990E-04
93	JUNC	43.28	142.6	0.	32.58	48.09	-3.108	0.	-5.4026E-04
92	JUNC	41.88	142.6	0.	32.64	47.25	-2.606	0.	-4.3344E-04
91	JUNC	40.76	142.6	0.	32.70	46.58	-2.082	0.	-3.6383E-04
6000	JUNC	-44.64	14.37	4.5776E-02	2.992	5.807	0.	1.1921E-07	1.5259E-05
1	JUNC	-44.91	14.27	0.2859	0.5113	49.45	366.3	0.	3.0518E-05
2	JUNC	-43.51	13.85	1.000	0.1426	180.2	49.80	0.	1.9073E-05
3	JUNC	-27.96	13.54	1.000	0.1335	185.8	17.48	0.	2.8610E-05
4	JUNC	-22.73	13.23	1.000	0.1286	187.8	6.056	0.	-3.6907E-04
1999	PLEN	80.00	143.7	1.000	1.353	212.3	0.	0.	0.

MODEL = SINDA85
FASTIC

SYSTEM REPRESENTATION OF DIRECT COND. HEAT PUMP RADIATOR

FLUID SUBMODEL NAME = COOL ; FLUID NO. = 290

LOOPCT = 133
CONVERGENCE STATUS = SUBMODEL CONVERGED AS OF 132 ITERATIONS. ENERGY STABLE BUT UNBALANCED

TIE PARAMETER TABULATION FOR SUBMODEL COOL

TIE	TYPE	UA	QTIE	LUMP	TEMP.	NODE	TEMP.	PATH 1	FRACT	PATH 2	FRACT
2001	RTU	0.	0.	2001	79.99	VERT.1	70.00				
2002	RTU	0.	0.	2002	79.99	VERT.2	70.00				
2003	RTU	0.	0.	2003	79.99	VERT.3	70.00				
2004	RTU	0.	0.	2004	79.98	VERT.4	70.00				
2005	RTU	0.	0.	2005	79.98	VERT.5	70.00				
81	RINC	12.57	-16.58	81	79.86	PLATE.81	78.54	81	1.0000	82	0.5000
82	RINC	3.618	-15.54	82	79.77	PLATE.82	75.48	82	0.5000	83	0.5000
83	RINC	2.499	-15.17	83	79.65	PLATE.83	73.57	83	0.5000	84	0.5000
84	RINC	1.483	-14.77	84	79.55	PLATE.84	69.59	84	0.5000	85	0.5000
85	RINC	0.3212	-8.962	85	78.26	PLATE.85	50.36	85	0.5000	86	0.5000
86	RINC	0.3423	-5.739	86	54.37	PLATE.86	37.61	86	0.5000	87	1.0000
96	RINC	0.9477	-7.867	96	50.19	PLATE.96	41.89	97	1.0000	96	0.5000
97	RINC	0.9213	-5.909	95	47.03	PLATE.95	40.62	96	0.5000	95	0.5000
98	RINC	0.9039	-3.874	94	44.95	PLATE.94	40.67	95	0.5000	94	0.5000
99	RINC	0.8899	-3.108	93	43.28	PLATE.93	39.79	94	0.5000	93	0.5000
100	RINC	0.8782	-2.606	92	41.88	PLATE.92	38.91	93	0.5000	92	0.5000
101	RINC	0.8688	-2.082	91	40.76	PLATE.91	38.36	92	0.5000	91	1.0000
1	RINC	14.70	366.3	1	-44.91	PLATE.1	-20.00	1	1.0000	2	0.5000
2	RINC	2.118	49.80	2	-43.51	PLATE.1	-20.00	2	0.5000	3	0.5000
3	RINC	2.196	17.48	3	-27.96	PLATE.1	-20.00	3	0.5000	4	0.5000
4	RINC	2.221	6.056	4	-22.73	PLATE.1	-20.00	4	0.5000	5	1.0000

MODEL = SINDA85
FASTIC

SYSTEM REPRESENTATION OF DIRECT COND. HEAT PUMP RADIATOR

FLUID SUBMODEL NAME = COOL ; FLUID NO. = 290

LOOPCT = 133
CONVERGENCE STATUS = SUBMODEL CONVERGED AS OF 132 ITERATIONS. ENERGY STABLE BUT UNBALANCED

TWINNED PATH PARAMETER TABULATION FOR SUBMODEL COOL

PATH	TWIN	TYPE	LMP 1	LMP 2	DUP I	DUP J	XL FLOW	FR TOTAL	DELTA PRES	FLOW REGIMES OR REYNOLDS NO	
2001H	2011X	TUBE	2000	2001	1.0	1.0	1.000	0.	4.3211E-03	0.	SLUG
2002H	2012X	TUBE	2001	2002	1.0	1.0	0.9550	0.	4.9139E-03	SLUG	SLUG
2003H	2013X	TUBE	2002	2003	1.0	1.0	0.9550	0.	4.9015E-03	SLUG	SLUG
2004H	2014X	TUBE	2003	2004	1.0	1.0	0.9600	0.	4.8770E-03	ANNULAR	ANNULAR
2005H	2015X	TUBE	2004	2005	1.0	1.0	0.9650	0.	4.8647E-03	SLUG	SLUG
81H	1081X	STUBE	108	81	8.0	1.0	1.000	0.3892	8.6089E-02	7458.1	
82L	1082V	STUBE	81	82	1.0	1.0	0.9800	0.3892	0.1755	ANNULAR	
83L	1083V	STUBE	82	83	1.0	1.0	0.6985	0.3892	0.2553	ANNULAR	
84L	1084V	STUBE	83	84	1.0	1.0	0.4239	0.3892	0.1931	ANNULAR	
85L	1085V	STUBE	84	85	1.0	1.0	0.1567	0.3892	2.2260E-02	SLUG	
86H	1086X	STUBE	85	86	1.0	1.0	0.	0.3892	1.6133E-02	665.54	
87H	1087X	STUBE	86	208	1.0	8.0	0.	0.3892	8.3660E-03	578.30	
1L	1001V	STUBE	6000	1	1.0	1.0	0.2695	3.113	9.6888E-02	ANNULAR	
2L	1002V	STUBE	1	2	1.0	1.0	0.9155	3.113	0.4199	ANNULAR	
3H	1003X	STUBE	2	3	1.0	1.0	1.000	3.113	0.3099	24496.	
4H	1004X	STUBE	3	4	1.0	1.0	1.000	3.113	0.3095	23658.	
5H	1005X	STUBE	4	1000	1.0	1.0	1.000	3.113	0.1556	23387.	

MODEL = SINDA85
FASTIC

SYSTEM REPRESENTATION OF DIRECT COND. HEAT PUMP RADIATOR

FLUID SUBMODEL NAME = COOL ; FLUID NO. = 290

LOOPCT = 133
CONVERGENCE STATUS = SUBMODEL CONVERGED AS OF 132 ITERATIONS. ENERGY STABLE BUT UNBALANCED

PATH PARAMETER TABULATION FOR SUBMODEL COOL

PATH	TYPE	LMP 1	LMP 2	DUP I	DUP J	STAT	XL UPSTRM	FLOWRATE	DELTA PRES	FLOW REGIMES OR REYNOLDS NO.	
2001H	TUBE	2000	2001	1.0	1.0	NORM	1.000	0.	4.3211E-03	0.	SLUG
2002H	TUBE	2001	2002	1.0	1.0	NORM	0.9550	0.	4.9139E-03	SLUG	SLUG
2003H	TUBE	2002	2003	1.0	1.0	NORM	0.9550	0.	4.9015E-03	SLUG	SLUG
2004H	TUBE	2003	2004	1.0	1.0	NORM	0.9600	0.	4.8770E-03	ANNULAR	ANNULAR
2005H	TUBE	2004	2005	1.0	1.0	NORM	0.9650	0.	4.8647E-03	SLUG	SLUG
1000	NULL	1000	2000	1.0	1.0	NORM	1.000	3.113	-130.6		
1999	STUBE	1999	2000	1.0	1.0	NORM	1.000	0.	1.0676E-09	0.	0.
108	STUBE	2000	108	1.0	1.0	NORM	1.000	3.113	0.1790	26102.	26103.
81H	STUBE	108	81	8.0	1.0	NORM	1.000	0.3892	8.6089E-02	7458.1	ANNULAR
82L	STUBE	81	82	1.0	1.0	DLS	0.	7.7898E-03	0.1755	13.449	
1082V	STUBE	81	82	1.0	1.0	DVS	1.000	0.3814	0.1755	8333.9	
83L	STUBE	82	83	1.0	1.0	DLS	0.	0.1173	0.2553	202.47	
1083V	STUBE	82	83	1.0	1.0	DVS	1.000	0.2718	0.2553	5941.0	
84L	STUBE	83	84	1.0	1.0	DLS	0.	0.2242	0.1931	386.57	
1084V	STUBE	83	84	1.0	1.0	DVS	1.000	0.1650	0.1931	3606.3	
85L	STUBE	84	85	1.0	1.0	DLS	0.	0.3282	2.2260E-02	565.57	
1085V	STUBE	84	85	1.0	1.0	DVS	1.000	6.0966E-02	2.2260E-02	1333.0	
86H	STUBE	85	86	1.0	1.0	NORM	0.	0.3892	1.6133E-02	665.54	578.30
87H	STUBE	86	208	1.0	8.0	NORM	0.	0.3892	8.3660E-03	578.30	578.30
208	STUBE	208	3000	1.0	1.0	NORM	0.	3.113	8.2954E-02	2698.7	2698.7
97	STUBE	3000	96	1.0	1.0	NORM	0.	3.113	1.2598E-02	2698.7	2634.0
96	STUBE	96	95	1.0	1.0	NORM	0.	3.113	2.4384E-02	2634.0	2586.4
95	STUBE	95	94	1.0	1.0	NORM	0.	3.113	2.3735E-02	2586.4	2555.6
94	STUBE	94	93	1.0	1.0	NORM	0.	3.113	2.3257E-02	2555.6	2531.2
93	STUBE	93	92	1.0	1.0	NORM	0.	3.113	2.2864E-02	2531.2	2511.0
92	STUBE	92	91	1.0	1.0	NORM	0.	3.113	2.2546E-02	2511.0	2494.9
91	STUBE	91	4000	1.0	1.0	NORM	0.	3.113	1.1212E-02	2494.9	2494.9
4000	STUBE	4000	5000	1.0	1.0	NORM	0.	3.113	0.1345	2494.9	2494.9
5000	LOSS	5000	6000	1.0	1.0	NORM	0.	3.113	128.1		
1L	STUBE	6000	1	1.0	1.0	DLS	0.	2.274	9.6888E-02	577.79	
1001V	STUBE	6000	1	1.0	1.0	DVS	1.000	0.8391	9.6888E-02	6618.8	
2L	STUBE	1	2	1.0	1.0	DLS	0.	0.2632	0.4199	66.793	
1002V	STUBE	1	2	1.0	1.0	DVS	1.000	2.850	0.4199	22497.	
3H	STUBE	2	3	1.0	1.0	NORM	1.000	3.113	0.3099	24496.	
4H	STUBE	3	4	1.0	1.0	NORM	1.000	3.113	0.3095	23658.	23658.
5H	STUBE	4	1000	1.0	1.0	NORM	1.000	3.113	0.1556	23387.	23387.
											23391.

MODEL = SINDA85
FASTIC

SYSTEM REPRESENTATION OF DIRECT COND. HEAT PUMP RADIATOR

++CAUTION++ ENERGY FLOW IN THERMAL SUBMODEL(S) IS STABLE BUT NOT BALANCED
TOTAL ENERGY INTO MODEL: 102.208
TOTAL ENERGY OUT OF MODEL: 114.889

SUBMODEL NAME = PLATE

ARLXCA =	1.000000E-02	ARLXCC =	1.01000	ATMPCA =	1.000000E+30	ATMPCC =	0.	BACKUP =	0.
CSGFAC =	0.	CSGMAX =	0.	CSGMIN =	0.	DRLXCA =	1.000000E-02	DRLXCC =	1.01000
DTIMEB =	1.000000E+30	DTIMEI =	0.	DTIMEL =	0.	DTIMEU =	0.	DTMPCA =	1.000000E+30
DTMPCC =	0.	EBALNA =	0.	EBALNC =	-2.613068E-04	EBALSA =	1.000000E-02	EBALSC =	1.000000E+30
ESUMIS =	0.	ESUMOS =	0.	EXTLIM =	50.0000	ITBOLD =	10	NARLXN =	0
NATMPN =	0	NCGMAN =	0	NCSGMN =	0	NDRLXN =	0	NDTMPN =	0
NEBALN =	0	NLOOPT =	0	ITEROT =	0	ITERXT =	3	OPEITR =	0
OUTPUT =	0.								

SUBMODEL NAME = VERT

ARLXCA =	1.000000E-02	ARLXCC =	1.01000	ATMPCA =	1.000000E+30	ATMPCC =	0.	BACKUP =	0.
CSGFAC =	0.	CSGMAX =	0.	CSGMIN =	0.	DRLXCA =	1.000000E-02	DRLXCC =	1.01000
DTIMEB =	1.000000E+30	DTIMEI =	0.	DTIMEL =	0.	DTIMEU =	0.	DTMPCA =	1.000000E+30
DTMPCC =	0.	EBALNA =	0.	EBALNC =	-3.814697E-06	EBALSA =	1.000000E-02	EBALSC =	1.000000E+30
ESUMIS =	0.	ESUMOS =	0.	EXTLIM =	50.0000	ITBOLD =	10	NARLXN =	0
NATMPN =	0	NCGMAN =	0	NCSGMN =	0	NDRLXN =	0	NDTMPN =	0
NEBALN =	0	NLOOPT =	0	ITEROT =	0	ITERXT =	3	OPEITR =	0
OUTPUT =	0.								

GLOBAL CONTROL CONSTANTS

ABSZRO =	-459.670	SIGMA =	1.713000E-09	TIMEM =	0.	TIMEN =	0.	TIMEND =	0.
TIMEO =	0.								

MODEL = SINDA85
FASTIC

SYSTEM REPRESENTATION OF DIRECT COND. HEAT PUMP RADIATOR

FLUID SUBMODEL NAME = COOL ; FLUID NO. = 290

LOOPCT = 18
CONVERGENCE STATUS = SUBMODEL CONVERGED AS OF 14 ITERATIONS. ENERGY STABLE BUT UNBALANCED

LUMP PARAMETER TABULATION FOR SUBMODEL COOL

LUMP	TYPE	TEMP	PRESSURE	QUALITY	DENSITY	ENTHALPY	HEAT RATE	MASS RATE	ENERGY RATE
1000	TANK	-22.78	13.08	1.000	0.1271	187.8	0.	0.	2.5635E-03
2001	TANK	79.99	143.7	0.9550	1.414	205.9	0.	0.	0.
2002	TANK	79.99	143.7	0.9550	1.414	205.9	0.	0.	0.
2003	TANK	79.99	143.7	0.9600	1.407	206.6	0.	0.	0.
2004	TANK	79.98	143.7	0.9650	1.400	207.3	0.	0.	0.
2005	TANK	79.98	143.7	0.9650	1.400	207.3	0.	0.	0.
5000	TANK	40.77	142.5	0.	32.70	46.59	0.	0.	-7.6294E-05
2000	JUNC	158.9	143.7	1.000	1.073	252.0	200.0	0.	0.
108	JUNC	158.9	143.5	1.000	1.071	252.0	0.	0.	0.
81	JUNC	79.87	143.5	0.9799	1.377	209.4	-16.59	0.	0.
82	JUNC	79.76	143.2	0.6987	1.895	169.5	-15.53	0.	-9.5367E-07
83	JUNC	79.65	143.0	0.4241	2.997	130.5	-15.17	0.	-2.5368E-04
84	JUNC	79.59	142.9	0.1566	6.952	92.55	-14.78	0.	9.5367E-07
85	JUNC	78.29	142.9	0.	30.80	69.52	-8.966	0.	8.7738E-05
86	JUNC	54.40	142.9	0.	32.03	54.77	-5.742	0.	9.5367E-07
208	JUNC	54.40	142.9	0.	32.03	54.77	0.	0.	1.7853E-03
3000	JUNC	54.40	142.8	0.	32.03	54.77	0.	0.	0.
4000	JUNC	40.77	142.6	0.	32.70	46.59	0.	0.	-9.1553E-05
96	JUNC	50.21	142.8	0.	32.24	52.24	-7.872	0.	5.2452E-06
95	JUNC	47.05	142.7	0.	32.39	50.34	-5.914	0.	3.8147E-06
94	JUNC	44.98	142.7	0.	32.50	49.10	-3.877	0.	-1.3399E-04
93	JUNC	43.30	142.7	0.	32.58	48.10	-3.113	0.	-8.6069E-04
92	JUNC	41.90	142.7	0.	32.64	47.26	-2.611	0.	-3.3951E-04
91	JUNC	40.77	142.7	0.	32.70	46.59	-2.085	0.	-1.5545E-04
6000	JUNC	-44.60	14.38	0.2695	0.5459	46.59	0.	0.	0.
1	JUNC	-45.21	14.17	0.9174	0.1598	164.5	367.2	0.	-3.0518E-05
2	JUNC	-43.25	13.85	1.000	0.1424	180.3	49.29	0.	2.2888E-03
3	JUNC	-27.87	13.55	1.000	0.1336	185.9	17.29	0.	9.5367E-06
4	JUNC	-22.70	13.24	1.000	0.1287	187.8	5.991	0.	4.7207E-04
1999	PLEN	80.00	143.7	1.000	1.353	212.3	0.	0.	0.

MODEL = SINDA85 SYSTEM REPRESENTATION OF DIRECT COND. HEAT PUMP RADIATOR
FASTIC

FLUID SUBMODEL NAME = COOL ; FLUID NO. = 290

LOOPCT = 18
CONVERGENCE STATUS = SUBMODEL CONVERGED AS OF 14 ITERATIONS. ENERGY STABLE BUT UNBALANCED

TIE PARAMETER TABULATION FOR SUBMODEL COOL

TIE	TYPE	UA	QTIE	LUMP	TEMP.	NODE	TEMP.	PATH 1	FRACT	PATH 2	FRACT
2001	BTU	0.	0.	2001	79.99	VERT.1	70.00				
2002	BTU	0.	0.	2002	79.99	VERT.2	70.00				
2003	BTU	0.	0.	2003	79.99	VERT.3	70.00				
2004	BTU	0.	0.	2004	79.98	VERT.4	70.00				
2005	BTU	0.	0.	2005	79.98	VERT.5	70.00				
81	BTNC	12.61	-16.59	81	79.87	PLATE.81	78.55	81	1.0000	82	0.5000
82	BTNC	3.620	-15.53	82	79.76	PLATE.82	75.47	82	0.5000	83	0.5000
83	BTNC	2.500	-15.17	83	79.65	PLATE.83	73.58	83	0.5000	84	0.5000
84	BTNC	1.483	-14.78	84	79.59	PLATE.84	69.63	84	0.5000	85	0.5000
85	BTNC	0.3212	-8.966	85	78.29	PLATE.85	50.38	85	0.5000	86	0.5000
86	BTNC	0.3423	-5.742	86	54.40	PLATE.86	37.62	86	0.5000	87	1.0000
96	BTNC	0.9485	-7.872	96	50.21	PLATE.96	41.91	97	1.0000	96	0.5000
97	BTNC	0.9220	-5.914	95	47.05	PLATE.95	40.64	96	0.5000	95	0.5000
98	BTNC	0.9047	-3.877	94	44.98	PLATE.94	40.69	95	0.5000	94	0.5000
99	BTNC	0.8907	-3.113	93	43.30	PLATE.93	39.81	94	0.5000	93	0.5000
100	BTNC	0.8789	-2.611	92	41.90	PLATE.92	38.93	93	0.5000	92	0.5000
101	BTNC	0.8695	-2.085	91	40.77	PLATE.91	38.38	92	0.5000	91	1.0000
1	BTNC	14.57	367.2	1	-45.21	PLATE.1	-20.00	1	1.0000	2	0.5000
2	BTNC	2.120	49.29	2	-43.25	PLATE.1	-20.00	2	0.5000	3	0.5000
3	BTNC	2.197	17.29	3	-27.87	PLATE.1	-20.00	3	0.5000	4	0.5000
4	BTNC	2.222	5.991	4	-22.70	PLATE.1	-20.00	4	0.5000	5	1.0000

MODEL = SINDA85 SYSTEM REPRESENTATION OF DIRECT COND. HEAT PUMP RADIATOR
FASTIC

FLUID SUBMODEL NAME = COOL ; FLUID NO. = 290

LOOPCT = 18
CONVERGENCE STATUS = SUBMODEL CONVERGED AS OF 14 ITERATIONS. ENERGY STABLE BUT UNBALANCED

TWINNED PATH PARAMETER TABULATION FOR SUBMODEL COOL

PATH	TWIN	TYPE	LMP 1	LMP 2	DUP I	DUP J	XL FLOW	FR TOTAL	DELTA PRES	FLOW REGIMES OR REYNOLDS NO	
2001H	2011X	TUBE	2000	2001	1.0	1.0	1.000	0.	4.3212E-03	0.	SLUG
2002H	2012X	TUBE	2001	2002	1.0	1.0	0.9550	0.	4.9139E-03	SLUG	SLUG
2003H	2013X	TUBE	2002	2003	1.0	1.0	0.9550	0.	4.9015E-03	ANNULAR	ANNULAR
2004H	2014X	TUBE	2003	2004	1.0	1.0	0.9600	0.	4.8770E-03	ANNULAR	ANNULAR
2005H	2015X	TUBE	2004	2005	1.0	1.0	0.9650	0.	4.8647E-03	ANNULAR	ANNULAR
81H	1081X	STUBE	108	81	8.0	1.0	1.000	0.3893	7.1974E-02	7460.4	
82H	1082X	STUBE	81	82	1.0	1.0	0.9799	0.3893	0.2152	ANNULAR	
83H	1083X	STUBE	82	83	1.0	1.0	0.6987	0.3893	0.2257	ANNULAR	
84H	1084X	STUBE	83	84	1.0	1.0	0.4241	0.3893	0.1175	ANNULAR	
85H	1085X	STUBE	84	85	1.0	1.0	0.1566	0.3893	1.3014E-02	SLUG	
86H	1086X	STUBE	85	86	1.0	1.0	0.	0.3893	1.5859E-02	665.85	
87H	1087X	STUBE	86	208	1.0	8.0	0.	0.3893	8.3674E-03	578.56	
1H	1001X	STUBE	6000	1	1.0	1.0	0.2695	3.114	0.2135	ANNULAR	
2H	1002X	STUBE	1	2	1.0	1.0	0.9174	3.114	0.3198	ANNULAR	
3H	1003X	STUBE	2	3	1.0	1.0	1.000	3.114	0.2970	24489.	
4H	1004X	STUBE	3	4	1.0	1.0	1.000	3.114	0.3095	23660.	
5H	1005X	STUBE	4	1000	1.0	1.0	1.000	3.114	0.1556	23392.	

MODEL = SINDA85 SYSTEM REPRESENTATION OF DIRECT COND. HEAT PUMP RADIATOR
FASTIC

FLUID SUBMODEL NAME = COOL ; FLUID NO. = 290

LOOPCT = 18
CONVERGENCE STATUS = SUBMODEL CONVERGED AS OF 14 ITERATIONS. ENERGY STABLE BUT UNBALANCED

PATH PARAMETER TABULATION FOR SUBMODEL COOL

PATH	TYPE	LMP 1	LMP 2	DUP I	DUP J	STAT	XL UPSTRM	FLOWRATE	DELTA PRES	FLOW REGIMES OR REYNOLDS NO.	
2001B	TUBE	2000	2001	1.0	1.0	NORM	1.000	0.	4.3212E-03	0.	SLUG
2002B	TUBE	2001	2002	1.0	1.0	NORM	0.9550	0.	4.9139E-03	SLUG	SLUG
2003B	TUBE	2002	2003	1.0	1.0	NORM	0.9550	0.	4.9015E-03	ANNULAR	ANNULAR
2004B	TUBE	2003	2004	1.0	1.0	NORM	0.9600	0.	4.8770E-03	ANNULAR	ANNULAR
2005B	TUBE	2004	2005	1.0	1.0	NORM	0.9650	0.	4.8647E-03	ANNULAR	ANNULAR
1000	NULL	1000	2000	1.0	1.0	NORM	1.000	3.114	-130.6		
1999	STUBE	1999	2000	1.0	1.0	NORM	1.000	0.	1.0676E-09	0.	0.
108	STUBE	2000	108	1.0	1.0	NORM	1.000	3.114	0.1791	26110.	26111.
81B	STUBE	108	81	8.0	1.0	NORM	1.000	0.3893	7.1974E-02	7460.4	ANNULAR
82B	STUBE	81	82	1.0	1.0	NORM	0.9799	0.3893	0.2152	ANNULAR	ANNULAR
83B	STUBE	82	83	1.0	1.0	NORM	0.6987	0.3893	0.2257	ANNULAR	ANNULAR
84B	STUBE	83	84	1.0	1.0	NORM	0.4241	0.3893	0.1175	ANNULAR	ANNULAR
85B	STUBE	84	85	1.0	1.0	NORM	0.1566	0.3893	1.3014E-02	SLUG	665.85
86B	STUBE	85	86	1.0	1.0	NORM	0.	0.3893	1.5859E-02	665.85	578.56
87B	STUBE	86	208	1.0	8.0	NORM	0.	0.3893	8.3674E-03	578.56	578.55
208	STUBE	208	3000	1.0	1.0	NORM	0.	3.114	8.3048E-02	2699.9	2699.9
97	STUBE	3000	96	1.0	1.0	NORM	0.	3.114	1.2615E-02	2699.9	2635.1
96	STUBE	96	95	1.0	1.0	NORM	0.	3.114	2.4418E-02	2635.1	2587.5
95	STUBE	95	94	1.0	1.0	NORM	0.	3.114	2.3769E-02	2587.5	2556.7
94	STUBE	94	93	1.0	1.0	NORM	0.	3.114	2.3290E-02	2556.7	2532.3
93	STUBE	93	92	1.0	1.0	NORM	0.	3.114	2.2896E-02	2532.3	2512.0
92	STUBE	92	91	1.0	1.0	NORM	0.	3.114	2.2576E-02	2512.0	2495.9
91	STUBE	91	4000	1.0	1.0	NORM	0.	3.114	1.1227E-02	2495.9	2495.9
4000	STUBE	4000	5000	1.0	1.0	NORM	0.	3.114	0.1347	2495.9	2495.9
5000	LOSS	5000	6000	1.0	1.0	NORM	0.	3.114	128.1		
1B	STUBE	6000	1	1.0	1.0	NORM	0.2695	3.114	0.2135	ANNULAR	ANNULAR
2B	STUBE	1	2	1.0	1.0	NORM	0.9174	3.114	0.3198	ANNULAR	24489.
3B	STUBE	2	3	1.0	1.0	NORM	1.000	3.114	0.2970	24489.	23660.
4B	STUBE	3	4	1.0	1.0	NORM	1.000	3.114	0.3095	23660.	23392.
5B	STUBE	4	1000	1.0	1.0	NORM	1.000	3.114	0.1556	23392.	23396.

++CAUTION++ ENERGY FLOW IN THERMAL SUBMODEL(S) IS STABLE BUT NOT BALANCED
TOTAL ENERGY INTO MODEL: 102.252
TOTAL ENERGY OUT OF MODEL: 114.929

MODEL = SINDA85 SYSTEM REPRESENTATION OF DIRECT COND. HEAT PUMP RADIATOR
NODMAP

NODE PLATE 91

A QMAP OF INPUT ARITH NODE PLATE 91 (INTERNAL 7)

THE PARAMETERS OF NODE PLATE 91 ARE:

TEMPERATURE	=	38.3755	(DEG.)
CAPACITANCE	=	0.	(ENERGY/DEG)
NET SOURCE/SINK	=	2.08467	(ENERGY/TIME, INCLUDES TIES)
CAP./SUM OF COND.	=	0.	(TIME, INCLUDES TIES)

THE ADJOINING NODES TO NODE PLATE 91 ARE:

NODE INPUT	(INTERNAL)	CONDUCTOR INPUT (INTERNAL)	TYPE	CONDUCTOR VALUE	% OF TYPE	% OF TOTAL	HEAT TRANSFER RATE (ENERGY/TIME)	TEMPERATURE OF ADJOINING NODE
PLATE	92(8)	91 (6)	LINEAR	5.000000E-02	30.8	14.9	2.753601E-02	38.9262
PLATE	81(1)	801 (11)	LINEAR	0.112500	69.2	33.5	4.51998	78.5531
PLATE	999(13)	991 (29)	RADIAT	0.229333	100.0	51.5	-6.63216	0.

THE ADJOINING LUMPS TO NODE PLATE 91 ARE:

LUMP MODEL.INPUT	(INTL)	TIE INPUT(INTL)	TYPE	DUPN	TIE CONDUCTANCE	HEAT RATE	ADJOINING LUMP PROPERTIES TEMPERATURE	DUPL
COOL.91	(24)	101(17)	HTNC	1.000	0.869516	2.08467	40.7729	1.000

THE TOTALS ON NODE PLATE 91 ARE:

LINEAR HEAT TRANSFER (CONDUCTION/CONVECTION)...	4.54751	
RADIATION HEAT TRANSFER.....	-6.63216	
TIE HEAT TRANSFER (FROM LUMPS).....	2.08467	
HEAT SOURCE/SINKS APPLIED.....	0.	
	3.051758E-05	(ENERGY/TIME)
EFFECTIVE ERN TEMPERATURE.....	0.	

MODEL = SINDA85 SYSTEM REPRESENTATION OF DIRECT COND. HEAT PUMP RADIATOR
STDSTL

FLUID SUBMODEL NAME = COOL ; FLUID NO. = 290

LOOPCT = 97
MAX TIME STEP (GROWTH LIMITED) = 5.663637E-02; LAST CAUSE: TUBE 2015; FLOWRATE CHANGE LIMIT (WAS 2.1605E-07)
LAST PSEUDO TIME STEP = 1.000000E-02 VS. RSMAXF = 1.000000E-02
CONVERGENCE STATUS = SUBMODEL CONVERGED AS OF 97 ITERATIONS. ENERGY STABLE BUT UNBALANCED

LUMP PARAMETER TABULATION FOR SUBMODEL COOL

LUMP	TYPE	TEMP	PRESSURE	QUALITY	DENSITY	ENTHALPY	HEAT RATE	MASS RATE	ENERGY RATE
1000	TANK	-22.76	13.00	1.000	0.1263	187.8	0.	-2.1458E-06	1.3428E-03
2001	TANK	79.72	143.2	0.9412	1.428	203.9	-0.7620	-1.9744E-07	-2.7657E-05
2002	TANK	79.72	143.2	0.9407	1.429	203.8	-0.7617	-2.6636E-07	-4.9829E-05
2003	TANK	79.71	143.2	0.9492	1.417	205.0	-0.7617	-2.7847E-07	-5.2214E-05
2004	TANK	79.71	143.1	0.9563	1.407	206.0	-0.7616	-2.2491E-07	-4.7684E-05
2005	TANK	79.71	143.1	0.9676	1.391	207.6	-0.7616	-2.8266E-07	-5.0306E-05
5000	TANK	40.49	142.0	0.	32.71	46.42	0.	2.8610E-06	-9.1095E-03
2000	JUNC	156.6	143.2	1.000	1.074	250.9	200.0	0.	0.
108	JUNC	156.6	143.0	1.000	1.073	250.9	0.	0.	0.
81	JUNC	79.59	142.9	0.9744	1.379	208.6	-16.46	5.9605E-08	5.7220E-06
82	JUNC	79.48	142.7	0.6940	1.899	168.7	-15.47	0.	1.9073E-06
83	JUNC	79.37	142.5	0.4204	3.009	129.9	-15.11	0.	9.5367E-07
84	JUNC	79.31	142.4	0.1540	7.020	92.03	-14.71	0.	-1.9073E-06
85	JUNC	77.70	142.3	0.	30.83	69.14	-8.895	2.9802E-08	9.5367E-07
86	JUNC	53.95	142.3	0.	32.05	54.50	-5.692	0.	0.
208	JUNC	53.95	142.3	0.	32.05	54.50	0.	0.	0.
3000	JUNC	53.95	142.2	0.	32.05	54.50	0.	0.	0.
4000	JUNC	40.48	142.1	0.	32.71	46.42	0.	-2.3842E-07	-1.5259E-05
96	JUNC	49.80	142.2	0.	32.26	51.99	-7.786	0.	1.9073E-06
95	JUNC	46.67	142.2	0.	32.41	50.11	-5.844	0.	3.3379E-06
94	JUNC	44.62	142.2	0.	32.51	48.88	-3.816	0.	-7.6294E-06
93	JUNC	42.97	142.2	0.	32.59	47.90	-3.058	0.	5.7220E-06
92	JUNC	41.59	142.1	0.	32.66	47.08	-2.564	0.	-9.5367E-07
91	JUNC	40.48	142.1	0.	32.71	46.42	-2.054	0.	1.9073E-06
6000	JUNC	-44.82	14.30	0.2691	0.5439	46.42	0.	0.	0.
1	JUNC	-45.43	14.09	0.9188	0.1588	164.7	367.7	0.	0.
2	JUNC	-43.08	13.77	1.000	0.1415	180.4	48.89	0.	0.
3	JUNC	-27.81	13.47	1.000	0.1327	185.9	17.14	0.	0.
4	JUNC	-22.68	13.16	1.000	0.1278	187.8	5.938	0.	0.
1999	PLEN	80.00	143.7	1.000	1.353	212.3	0.	-0.1265	-26.86

MODEL = SINDA85 SYSTEM REPRESENTATION OF DIRECT COND. HEAT PUMP RADIATOR
STDSTL

FLUID SUBMODEL NAME = COOL ; FLUID NO. = 290

LOOPCT = 97
MAX TIME STEP (GROWTH LIMITED) = 5.663637E-02; LAST CAUSE: TUBE 2015; FLOWRATE CHANGE LIMIT (WAS 2.1605E-07)
LAST PSEUDO TIME STEP = 1.000000E-02 VS. RSMAXF = 1.000000E-02
CONVERGENCE STATUS = SUBMODEL CONVERGED AS OF 97 ITERATIONS. ENERGY STABLE BUT UNBALANCED

TIE PARAMETER TABULATION FOR SUBMODEL COOL

TIE	TYPE	UA	QTIE	LUMP	TEMP.	NODE	TEMP.	PATH 1	FRACT	PATH 2	FRACT
2001	HTU	63.87	-0.7620	2001	79.72	VERT.1	79.71				
2002	HTU	63.20	-0.7617	2002	79.72	VERT.2	79.70				
2003	HTU	74.51	-0.7617	2003	79.71	VERT.3	79.70				
2004	HTU	87.14	-0.7616	2004	79.71	VERT.4	79.70				
2005	HTU	118.9	-0.7616	2005	79.71	VERT.5	79.70				
81	HTNC	11.04	-16.46	81	79.59	PLATE.81	78.10	81	1.0000	82	0.5000
82	HTNC	3.599	-15.47	82	79.48	PLATE.82	75.19	82	0.5000	83	0.5000
83	HTNC	2.486	-15.11	83	79.37	PLATE.83	73.30	83	0.5000	84	0.5000
84	HTNC	1.470	-14.71	84	79.31	PLATE.84	69.31	84	0.5000	85	0.5000
85	HTNC	0.3217	-8.895	85	77.70	PLATE.85	50.05	85	0.5000	86	0.5000
86	HTNC	0.3427	-5.692	86	53.95	PLATE.86	37.34	86	0.5000	87	1.0000
96	HTNC	0.9419	-7.786	96	49.80	PLATE.96	41.53	97	1.0000	96	0.5000
97	HTNC	0.9158	-5.844	95	46.67	PLATE.95	40.29	96	0.5000	95	0.5000
98	HTNC	0.8986	-3.816	94	44.62	PLATE.94	40.38	95	0.5000	94	0.5000
99	HTNC	0.8849	-3.058	93	42.97	PLATE.93	39.52	94	0.5000	93	0.5000
100	HTNC	0.8733	-2.564	92	41.59	PLATE.92	38.66	93	0.5000	92	0.5000
101	HTNC	0.8640	-2.054	91	40.48	PLATE.91	38.11	92	0.5000	91	1.0000
1	HTNC	14.46	367.7	1	-45.43	PLATE.1	-20.00	1	1.0000	2	0.5000
2	HTNC	2.118	48.89	2	-43.08	PLATE.1	-20.00	2	0.5000	3	0.5000
3	HTNC	2.194	17.14	3	-27.81	PLATE.1	-20.00	3	0.5000	4	0.5000
4	HTNC	2.219	5.938	4	-22.68	PLATE.1	-20.00	4	0.5000	5	1.0000

MODEL = SINDA85 SYSTEM REPRESENTATION OF DIRECT COND. HEAT PUMP RADIATOR
STDSTL

FLUID SUBMODEL NAME = COOL ; FLUID NO. = 290

LOOPCT = 97
MAX TIME STEP (GROWTH LIMITED) = 5.663637E-02; LAST CAUSE: TUBE 2015; FLOWRATE CHANGE LIMIT (WAS 2.1605E-07)
LAST PSEUDO TIME STEP = 1.000000E-02 VS. RSMAXF = 1.000000E-02
CONVERGENCE STATUS = SUBMODEL CONVERGED AS OF 97 ITERATIONS. ENERGY STABLE BUT UNBALANCED

TWINNED PATH PARAMETER TABULATION FOR SUBMODEL COOL

PATH	TWIN	TYPE	LMP 1	LMP 2	DUP I	DUP J	XL FLOW	FR TOTAL	DELTA PRES	FLOW REGIMES OR REYNOLDS NO	
2001L	2011V	TUBE	2000	2001	1.0	1.0	0.5000	-1.2498E-06	4.8486E-03	3.36498E-03	ANNULAR
2002L	2012V	TUBE	2001	2002	1.0	1.0	0.5000	-1.0524E-06	4.8497E-03	ANNULAR	ANNULAR
2003L	2013V	TUBE	2002	2003	1.0	1.0	0.5000	-7.8604E-07	4.8286E-03	ANNULAR	ANNULAR
2004L	2014V	TUBE	2003	2004	1.0	1.0	0.5000	-5.0757E-07	4.8144E-03	ANNULAR	ANNULAR
2005L	2015V	TUBE	2004	2005	1.0	1.0	0.5000	-2.8266E-07	4.7864E-03	ANNULAR	ANNULAR
81B	1081X	STUBE	108	81	8.0	1.0	1.000	0.3886	7.3768E-02	7474.4	
82B	1082X	STUBE	81	82	1.0	1.0	0.9744	0.3886	0.2191	ANNULAR	
83B	1083X	STUBE	82	83	1.0	1.0	0.6940	0.3886	0.2244	ANNULAR	
84B	1084X	STUBE	83	84	1.0	1.0	0.4204	0.3886	0.1162	ANNULAR	
85B	1085X	STUBE	84	85	1.0	1.0	0.1540	0.3886	1.3111E-02	SLUG	
86B	1086X	STUBE	85	86	1.0	1.0	0.	0.3886	1.5869E-02	662.42	
87B	1087X	STUBE	86	208	1.0	8.0	0.	0.3886	8.3698E-03	576.10	
1B	1001X	STUBE	6000	1	1.0	1.0	0.2691	3.109	0.2141	ANNULAR	
2B	1002X	STUBE	1	2	1.0	1.0	0.9188	3.109	0.3206	ANNULAR	
3B	1003X	STUBE	2	3	1.0	1.0	1.000	3.109	0.2981	24440.	
4B	1004X	STUBE	3	4	1.0	1.0	1.000	3.109	0.3106	23619.	
5B	1005X	STUBE	4	1000	1.0	1.0	1.000	3.109	0.1561	23353.	

MODEL = SINDA85 SYSTEM REPRESENTATION OF DIRECT COND. HEAT PUMP RADIATOR
STDSTL

FLUID SUBMODEL NAME = COOL ; FLUID NO. = 290

LOOPCT = 97
MAX TIME STEP (GROWTH LIMITED) = 5.663637E-02; LAST CAUSE: TUBE 2015; FLOWRATE CHANGE LIMIT (WAS 2.1605E-07)
LAST PSEUDO TIME STEP = 1.000000E-02 VS. RSMAXF = 1.000000E-02
CONVERGENCE STATUS = SUBMODEL CONVERGED AS OF 97 ITERATIONS. ENERGY STABLE BUT UNBALANCED

PATH PARAMETER TABULATION FOR SUBMODEL COOL

PATH	TYPE	LMP 1	LMP 2	DUP I	DUP J	STAT	XL UPSTRM	FLOWRATE	DELTA PRES	FLOW REGIMES OR REYNOLDS NO.	
2001L	TUBE	2000	2001	1.0	1.0	DLS	0.	-2.1102E-02	4.8486E-03		5.0960
2011V	TUBE	2000	2001	1.0	1.0	DVS	1.000	2.1101E-02	4.8486E-03	56.811	
2002L	TUBE	2001	2002	1.0	1.0	DLS	0.	-2.1486E-02	4.8497E-03		5.1888
2012V	TUBE	2001	2002	1.0	1.0	DVS	1.000	2.1485E-02	4.8497E-03	65.747	
2003L	TUBE	2002	2003	1.0	1.0	DLS	0.	-1.6114E-02	4.8286E-03		3.8914
2013V	TUBE	2002	2003	1.0	1.0	DVS	1.000	1.6114E-02	4.8286E-03	49.309	
2004L	TUBE	2003	2004	1.0	1.0	DLS	0.	-1.0742E-02	4.8144E-03		2.5941
2014V	TUBE	2003	2004	1.0	1.0	DVS	1.000	1.0742E-02	4.8144E-03	32.871	
2005L	TUBE	2004	2005	1.0	1.0	DLS	0.	-5.3713E-03	4.7864E-03		1.2971
2015V	TUBE	2004	2005	1.0	1.0	DVS	1.000	5.3710E-03	4.7864E-03	16.436	
1000	NULL	1000	2000	1.0	1.0	NORM	1.000	3.109	-130.2		
1999	STUBE	1999	2000	1.0	0.	NORM	1.000	0.1265	0.5491	4837.6	4258.4
108	STUBE	2000	108	1.0	1.0	NORM	1.000	3.109	0.1782	26159.	26160.
81B	STUBE	108	81	8.0	1.0	NORM	1.000	0.3886	7.3768E-02	7474.4	ANNULAR
82B	STUBE	81	82	1.0	1.0	NORM	0.9744	0.3886	0.2191	ANNULAR	ANNULAR
83B	STUBE	82	83	1.0	1.0	NORM	0.6940	0.3886	0.2244	ANNULAR	ANNULAR
84B	STUBE	83	84	1.0	1.0	NORM	0.4204	0.3886	0.1162	ANNULAR	ANNULAR
85B	STUBE	84	85	1.0	1.0	NORM	0.1540	0.3886	1.3111E-02	SLUG	662.42
86B	STUBE	85	86	1.0	1.0	NORM	0.	0.3886	1.5869E-02	662.42	576.10
87B	STUBE	86	208	1.0	8.0	NORM	0.	0.3886	8.3698E-03	576.10	576.10
208	STUBE	208	3000	1.0	1.0	NORM	0.	3.109	8.2328E-02	2688.5	2688.5
97	STUBE	3000	96	1.0	1.0	NORM	0.	3.109	1.2492E-02	2688.5	2624.6
96	STUBE	96	95	1.0	1.0	NORM	0.	3.109	2.4173E-02	2624.6	2577.6
95	STUBE	95	94	1.0	1.0	NORM	0.	3.109	2.3531E-02	2577.6	2547.3
94	STUBE	94	93	1.0	1.0	NORM	0.	3.109	2.3061E-02	2547.3	2523.4
93	STUBE	93	92	1.0	1.0	NORM	0.	3.109	2.2676E-02	2523.4	2503.5
92	STUBE	92	91	1.0	1.0	NORM	0.	3.109	2.2363E-02	2503.5	2487.7
91	STUBE	91	4000	1.0	1.0	NORM	0.	3.109	1.1122E-02	2487.7	2487.7
4000	STUBE	4000	5000	1.0	1.0	NORM	0.	3.109	0.1335	2487.7	2487.8
5000	LOSS	5000	6000	1.0	1.0	NORM	0.	3.109	127.7		
1B	STUBE	6000	1	1.0	1.0	NORM	0.2691	3.109	0.2141	ANNULAR	ANNULAR
2B	STUBE	1	2	1.0	1.0	NORM	0.9188	3.109	0.3206	ANNULAR	24440.
3B	STUBE	2	3	1.0	1.0	NORM	1.000	3.109	0.2981	24440.	23619.
4B	STUBE	3	4	1.0	1.0	NORM	1.000	3.109	0.3106	23619.	23353.
5B	STUBE	4	1000	1.0	1.0	NORM	1.000	3.109	0.1561	23353.	23357.

++CAUTION++ ENERGY FLOW IN THERMAL SUBMODEL(S) IS STABLE BUT NOT BALANCED
TOTAL ENERGY INTO MODEL: 105.254
TOTAL ENERGY OUT OF MODEL: 117.927

MODEL = SINDA85 SYSTEM REPRESENTATION OF DIRECT COND. HEAT PUMP RADIATOR

FLUID SUBMODEL NAME = COOL ; FLUID NO. = 290

LOOPCT = 97
MAX TIME STEP (GROWTH LIMITED) = 5.663637E-02; LAST CAUSE: TUBE 2015; FLOWRATE CHANGE LIMIT (WAS 2.1605E-07)
LAST PSEUDO TIME STEP = 1.000000E-02 VS. RSMAXF = 1.000000E-02
CONVERGENCE STATUS = SUBMODEL CONVERGED AS OF 97 ITERATIONS. ENERGY STABLE BUT UNBALANCED

LUMP EXTRA PARAMETER TABULATION FOR SUBMODEL COOL
BODY FORCE ACCELERATION COMPONENTS (X,Y,Z): 0. , 0. , 4.17312E+08

LUMP	TYPE	CX	CY	CZ	NO. PATHS	HOLDS	MASS	VOLUME	COMPLIANCE	VOLUME RATE
1000	TANK	0.	0.	0.	3	NONE	1.2629E-03	1.0000E-02	0.	0.
2001	TANK	0.	0.	0.5000	4	NONE	2.4343E-04	1.7044E-04	0.	0.
2002	TANK	0.	0.	1.000	4	NONE	2.4357E-04	1.7044E-04	0.	0.
2003	TANK	0.	0.	1.500	4	NONE	2.4146E-04	1.7044E-04	0.	0.
2004	TANK	0.	0.	2.000	4	NONE	2.3976E-04	1.7044E-04	0.	0.
2005	TANK	0.	0.	2.500	2	NONE	2.3707E-04	1.7044E-04	0.	0.
5000	TANK	0.	0.	0.	2	NONE	3.2712E-02	1.0000E-03	0.	0.
2000	JUNC	0.	0.	0.	5	NONE				
108	JUNC	0.	0.	0.	3	NONE				
81	JUNC	0.	0.	0.	4	NONE				
82	JUNC	0.	0.	0.	4	NONE				
83	JUNC	0.	0.	0.	4	NONE				
84	JUNC	0.	0.	0.	4	NONE				
85	JUNC	0.	0.	0.	4	NONE				
86	JUNC	0.	0.	0.	4	NONE				
208	JUNC	0.	0.	0.	3	NONE				
3000	JUNC	0.	0.	0.	2	NONE				
4000	JUNC	0.	0.	0.	2	NONE				
96	JUNC	0.	0.	0.	2	NONE				
95	JUNC	0.	0.	0.	2	NONE				
94	JUNC	0.	0.	0.	2	NONE				
93	JUNC	0.	0.	0.	2	NONE				
92	JUNC	0.	0.	0.	2	NONE				
91	JUNC	0.	0.	0.	2	NONE				
6000	JUNC	0.	0.	0.	3	NONE				
1	JUNC	0.	0.	0.	4	NONE				
2	JUNC	0.	0.	0.	4	NONE				
3	JUNC	0.	0.	0.	4	NONE				
4	JUNC	0.	0.	0.	4	NONE				
1999	PLEN	0.	0.	0.	1					

MODEL = SINDA85 SYSTEM REPRESENTATION OF DIRECT COND. HEAT PUMP RADIATOR

FLUID SUBMODEL NAME = COOL ; FLUID NO. = 290

LOOPCT = 97
MAX TIME STEP (GROWTH LIMITED) = 5.663637E-02; LAST CAUSE: TUBE 2015; FLOWRATE CHANGE LIMIT (WAS 2.1605E-07)
LAST PSEUDO TIME STEP = 1.000000E-02 VS. RSMAXF = 1.000000E-02
CONVERGENCE STATUS = SUBMODEL CONVERGED AS OF 97 ITERATIONS. ENERGY STABLE BUT UNBALANCED

(S)TUBE PARAMETER TABULATION FOR SUBMODEL COOL

PATH	T/S	DH	AF	TLEN	MRF	FC	FPOW	UPF	AC	EC	FK	IP
2001L	T	2.0833E-02	3.4088E-04	0.5000	0.	-8.9089E+05	0.000	1.000	-7.006	0.	0.	6
2011V	T	2.0833E-02	3.4088E-04	0.5000	0.	0.	0.000	1.000	-6.6989E+04	0.	0.	6
2002L	T	2.0833E-02	3.4088E-04	0.5000	0.	-8.9092E+05	0.000	1.000	86.86	0.	0.	6
2012V	T	2.0833E-02	3.4088E-04	0.5000	0.	0.	0.000	1.000	6.8932E+05	0.	0.	6
2003L	T	2.0833E-02	3.4088E-04	0.5000	0.	-8.9695E+05	0.000	1.000	179.2	0.	0.	6
2013V	T	2.0833E-02	3.4088E-04	0.5000	0.	0.	0.000	1.000	2.1243E+06	0.	0.	6
2004L	T	2.0833E-02	3.4088E-04	0.5000	0.	-8.9636E+05	0.000	1.000	255.4	0.	0.	6
2014V	T	2.0833E-02	3.4088E-04	0.5000	0.	0.	0.000	1.000	3.1868E+06	0.	0.	6
2005L	T	2.0833E-02	3.4088E-04	0.5000	0.	-9.1264E+05	0.000	1.000	186.5	0.	0.	6
2015V	T	2.0833E-02	3.4088E-04	0.5000	0.	0.	0.000	1.000	3.9846E+06	0.	0.	6
1999	S	1.6667E-03	2.1817E-06	1.000	0.	-1.0818E+12	0.689	0.500	0.	0.	0.	6
108	S	6.6667E-03	3.4907E-05	1.350	0.	-2.4960E+09	0.746	0.500	7.6512E+08	0.	0.	6
81H	S	2.9167E-03	6.6813E-06	0.2500	0.	-2.4027E+10	0.634	0.500	4.6444E+09	0.	0.	6
82H	S	2.9167E-03	6.6813E-06	0.5000	0.	-3.5582E+10	0.000	0.500	4.4451E+09	0.	0.	6
83H	S	2.9167E-03	6.6813E-06	0.5000	0.	-7.2304E+10	0.727	0.500	4.3509E+09	0.	0.	6
84H	S	2.9167E-03	6.6813E-06	0.5000	0.	-4.5936E+10	0.901	0.500	4.2527E+09	0.	0.	6
85H	S	2.9167E-03	6.6813E-06	0.5000	0.	-2.9834E+09	0.000	0.500	2.4645E+09	0.	0.	6
86H	S	2.9167E-03	6.6813E-06	0.5000	0.	-2.4624E+09	0.000	0.500	2.7802E+07	0.	0.	6
87H	S	2.9167E-03	6.6813E-06	0.2500	0.	-1.2931E+09	0.000	0.500	0.	0.	0.	6
208	S	5.0000E-03	1.9635E-05	1.350	0.	-4.3044E+08	1.000	0.500	-8.0919E+07	0.	0.	6
97	S	5.0000E-03	1.9635E-05	0.2500	0.	-7.8106E+07	1.000	0.500	5.1450E+05	0.	0.	6
96	S	5.0000E-03	1.9635E-05	0.5000	0.	-1.5053E+08	1.000	0.500	3.7977E+05	0.	0.	6
95	S	5.0000E-03	1.9635E-05	0.5000	0.	-1.4640E+08	1.000	0.500	2.4509E+05	0.	0.	6
94	S	5.0000E-03	1.9635E-05	0.5000	0.	-1.4343E+08	1.000	0.500	1.9474E+05	0.	0.	6
93	S	5.0000E-03	1.9635E-05	0.5000	0.	-1.4101E+08	1.000	0.500	1.6218E+05	0.	0.	6
92	S	5.0000E-03	1.9635E-05	0.5000	0.	-1.3903E+08	1.000	0.500	1.2922E+05	0.	0.	6
91	S	5.0000E-03	1.9635E-05	0.2500	0.	-6.9080E+07	1.000	0.500	0.	0.	0.	6
4000	S	5.0000E-03	1.9635E-05	3.000	0.	-8.2898E+08	1.000	0.500	0.	0.	0.	6
1H	S	1.0417E-02	8.5221E-05	0.7500	0.	-1.1696E+09	0.567	0.500	-6.1414E+08	0.	0.	6
2H	S	1.0417E-02	8.5221E-05	1.500	0.	-2.6731E+09	0.692	0.500	-1.0573E+08	0.	0.	6
3H	S	1.0417E-02	8.5221E-05	1.500	0.	-2.4195E+09	0.733	0.500	-5.4432E+07	0.	0.	6
4H	S	1.0417E-02	8.5221E-05	1.500	0.	-2.3995E+09	0.789	0.500	-3.9678E+07	0.	0.	6
5H	S	1.0417E-02	8.5221E-05	0.7500	0.	-1.2753E+09	0.759	0.500	0.	0.	0.	6

PROBLEM H: COMPARISONS WITH CLOSED FORM SOLUTION

This problem analyzes a thermal/fluid transient event for which a closed form solution exists. In addition to validating the code, this problem was chosen to demonstrate the following FLUINT features:

1. lumped parameter (finite difference) ties versus segment–oriented ties
2. axial resolution requirements for adequate accuracy
3. user logic for specialized initial and boundary conditions

H.1 Problem Description

A one inch diameter pipe carries liquid water flowing at 1000 lb_m/hr over a distance of ten feet (see Figure H–1). The pipe wall temperature is constant at 200°F. Dr. R. K. McMordie demonstrated that if both the initial axial temperature profile and the inlet temperature profile obeyed certain exponential relationships in space and time, respectively, and if most properties and coefficients could be assumed constant, then the energy equation was separable and a closed form 1D solution for the fluid temperature as a function of location and time, T(x,t), could be found.

The inlet temperature is determined by the following relationship in time:

$$T_i(t) = T_w \cdot e^{-(t/a)} \qquad \text{(H–1)}$$

$$a = \frac{\varrho \cdot D \cdot C_p}{4 \cdot U}$$

where T_w is the wall temperature, a is the time constant, ϱ is the density, D the diameter, C_p is the specific heat, and U is the heat transfer coefficient.

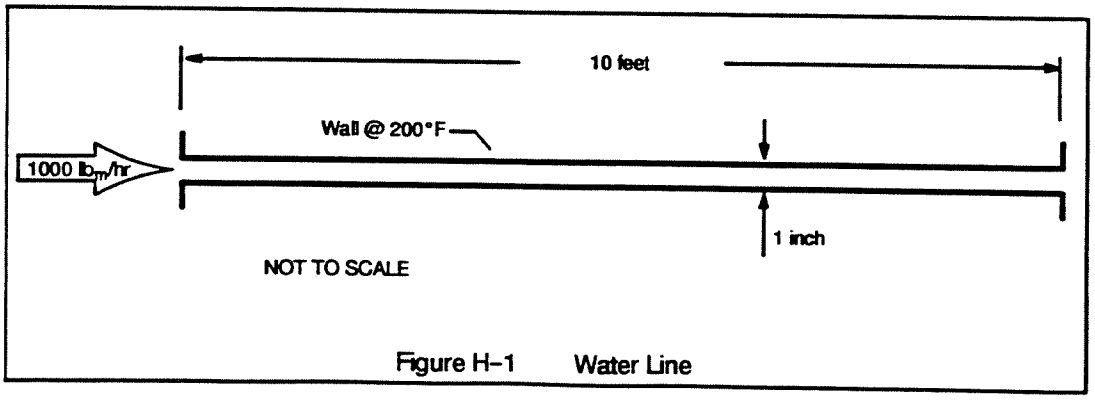


Figure H–1 Water Line

The initial temperature profile is a function of distance from the inlet:

$$T_o(x) = T_w \cdot (2 - e^{-(x/\beta)}) \qquad \text{(H-2)}$$

$$\beta = \frac{\dot{m} \cdot C_p}{\pi \cdot D \cdot U}$$

where β is the characteristic length, and $\dot{m}$ is the mass flowrate.

Under these circumstances, the 1D analytic solution is:

$$T(x,t) = T_w \cdot (1 - e^{-(x/\beta)}) + T_w \cdot e^{-(t/a)} \qquad \text{(H-3)}$$

In summary, the relevant characteristics of the problem are:

Line:		
	Diameter	1 inch
	Length	10 feet
Fluid:		
	Density	62.4 lb_m/ft^3
	Specific Heat	1.0 BTU/lb_m-°F
	Flowrate	1000 lb_m/hr
Boundary condition:		
	Wall temperature	200°F
	Head transfer coefficient	242 BTU/hr-ft^2-°F
	Inlet temperature profile	equation H-1
Initial condition:		
	Axial temperature profile	equation H-2

Assuming a 1D velocity profile, the time it takes for the fluid to move through the entire length of the line is about 12.3 seconds. Therefore, the transient event duration will be longer than this time, say 15 seconds, in order to provide a good basis of comparison.

H.2 A FLUINT Model

At least one modeling choices is clear: the line itself must be modeled with tanks to capture the transient movement of liquid. Any path may be used, since flowrate and density are both constant and pressure drop is irrelevant. The thermal boundary condition will be represented by a boundary node, and the flowrate boundary condition will be represented by an MFRSET connector. Also, it seems straight forward to use a plenum for the inlet state. The temperature of the inlet can be varied in FLOGIC 0 in a step–wise fashion as long as the time step is constrained such that this is a reasonable approximation. For the purist, a tank could similarly have been used, with the QDOT calculated such that the correct time derivative, dT/dt, was also imposed.

Other choices are less clear, and are mainly driven by the need to compare with a closed form solution that relies on simplifying assumptions: FLUINT must be prevented from including higher order effects, which it would by default. For example, using the default library description of water (or even one generated by RAPPR) would not only result in slight differences in density and C_p, it would include the variation of both of those parameters with temperature. As

the liquid in the pipe cools due to the influx of colder liquid, it would shrink and therefore cause the outlet flowrate to differ from the prescribed 1000 lb_m/hr by as much as 3%. Therefore, a very simple 9000 series fluid description must be used in which both the density and specific heat are constants.

Similarly, using convection ties (HTN, HTNC, HTNS) would violate the comparison because the calculated values of the heat transfer coefficients would be slightly different from 242 BTU/hr^2–°F, and would also vary with temperature. Therefore, user–defined ties (HTU and HTUS) must be used to hold the heat transfer coefficient constant.

Two decisions remain:

1. How many lumps are required to provide adequate accuracy?

2. Should the heat transfer be modeled using lumped parameter/finite difference methods (HTU ties) or as a collection of heat transfer segments (HTUS ties)?

These questions, which turn out to be interrelated, are the focus of this model and of the rest of this subsection.

The question of how many lumps are needed to adequately capture the transient event can be addressed to some extent by rules–of–thumb, the reliance on which is always somewhat dangerous. The first rule of thumb is that of length to diameter ratio: a good breakdown will have a L/D ratio for each element on the order of 10 (which is liberally construed to mean from 3 to 30). In this model, this advice would result in the use of about 10 lumps. The second rule of thumb for single–phase flow and conductive thermal systems in general is that temperature differences between nodes should not be greater than about 10°F. In this example problem, the range of temperatures in the problem are expected to be on the order of 100°F, so this rule–of–thumb concurs with the guess of 10 sections.

To envelope this guess and test its sensitivity, at least three models will be built using 5, 10, and 20 sections. While a real analyst would probably have rerun the same problem three times with slightly different inputs, for illustration purposes all models will be run simultaneously using different submodels for each discretization. These three submodels will use the standard HTU or lumped parameter style user–defined tie, which assumes that the lump and node each represent the average state of the fluid and wall, respectively, for each section. With these methods, there are no "hidden" temperature gradients within each section, and the user must use enough sections to capture any anticipated gradients in temperature or other properties. Too few sections will result in a slight underprediction of the heat transfer rates.

The lumped parameter HTU tie can be contrasted with the segment–oriented HTUS tie, which assumes that the endpoint lumps on each path represent distinct inlet and outlet fluid states, and that the node represents the average wall temperature over that entire length. Temperature gradients internal to the segment are presumed to follow an exponential relationship (e.g., LMTD methods are employed), enabling the use of far fewer lumps without sacrificing accuracy in single–phase flows. To compare this approach with the above submodels, a fourth submodel is added that uses HTUS ties instead of HTU ties, and breaks the line into only five sections.

All models use downstream discretized LINE macros. Part of the reason for using downstream lumps (as opposed to upstream or centered) is to better enable comparisons between HTU and HTUS ties, which are otherwise intrinsically difficult to substitute for each other.

The other reason for using downstream methods is more subtle, and demands an answer to the question: What axial coordinate, x, should be attributed to each lump? In other words, if Tank #1 models the first foot of the line, should it be assigned x=0.0 ft, x=0.5 ft, or x=1.0 ft? For the segment–oriented model, the lumps have defined locations corresponding to the inlet and outlets of each path length. For the lumped parameter models, the answer is less definitive.

Normally, the user need not be concerned with the following distinctions; where exactly a lump resides is largely immaterial. In fact, the user normally should not attempt to define locations to a tighter degree than the resolution of the model itself, which becomes the limiting uncertainty. However, to be able to compare with a continuous, analytic solution, and to calculate the correct initial conditions, the better understanding is necessary.

From the perspective of hydrodynamics (e.g., pressure profiles, etc.), the lump location is defined by the lengths of the paths adjacent to it. In other words, the first lump in the 10 tank model is located at x=0.0 ft for upstream discretization, x=0.5 ft for centered, and x=1.0 ft for downstream. On the other hand, from a thermal and heat transfer perspective, the lump always 'resides' at the current downstream end of the section it represents. Thus, while these two locations are coincident for downstream discretization, they are different for the other methods.

In this model, where hydrodynamics are irrelevant and temperature profiles are paramount, the axial coordinates of the lumps will be taken to be at the downstream end of each section—coincident with the segment–oriented approach. Initial conditions and analytic comparisons will be performed consistent with this convention.

H.3 The Input File

The input file is listed immediately following the last section.

OPTIONS DATA—Two user files are named, one to contain a summary of the temperature responses for all submodels and one to contain the temperature difference between the closed form solution and the FLUINT predictions. The DEBUG feature is turned off globally.

USER DATA—Values that will be used in several logic blocks are either initialized (π), or space for them is reserved (α and β).

NODE DATA—The wall temperature is set.

OPERATIONS DATA—All submodels are built, and headers are written to the user files. FASTIC is called to initialize the pressure profile (although it is largely irrelevant), and the initial spatial temperature profile is established using calls to CHGLMP. Next, the transient integration is performed using FORWRD, and the summary user files are written.

CONTROL DATA—English units are chosen, with units of °F and psia. The values of OUTPUT and OUTPTF are required, but are set larger than the solution time, resulting in OUTPUT CALLS only before and after the transient. The transient duration (TIMEND) is set to 15 seconds.

FLOW DATA—Four fluid submodels are built. Three are built using lumped parameter heat transfer methods and varying spatial discretization: LINE20, LINE10, and LINE5. A variation of LINE5 is written using segment–oriented methods: LINE5S. The working fluid is the same for all submodels: 9718, meaning a simple incompressible water description that is provided later in the input file.

The names and types of the elements are as follows:

PLENUM 100	inlet condition
TANKs 1–N	tanks in the line, numbered from the inlet
STUBEs 1–N	paths in the line
MFRSET 100	flowrate boundary condition at the end of the line
HTU(S) 1–N	heat transfer ties to node WALL.1

The lines are generated using downstream–discretized LINE macros, and the ties are generated with the GENT macro. The values of pressure are arbitrary.

FLOGIC 0—The inlet states (plenum temperatures) for all submodels are reset according to the prescribed profile in time. This same logic could have been placed in the FLOGIC block of any of the fluid submodels.

OUTPUT CALLS—Lump and tie tabulation routines are called for all submodels.

FPROP DATA—A very simple liquid water description is input, the purpose of which is to assure that density and specific heat stay constant.

H.4 Output Description

The user summary files and selected printings from the OUTPUT file are listed following the input file.

The initial temperature profile for all models along with the final temperature profiles for each model are all plotted in Figure H–2. The match is very good for all models, even the relatively coarse 5 tank submodel LINE5. For the rest of the models, the differences are nearly indistinguishable at this scale. As will be shown later, the maximum error in any submodel at any location was 3%, which is well within the noise range of engineering uncertainties. For the 20 tank lumped–parameter model (LINE20) and the 5 tank segment–oriented model (LINE5S), the match is excellent.

To better visualize the comparison between the closed form solution and the predictions, the difference in temperature between the closed form solution and the FLUINT predictions for each model are plotted in Figure H–3. For all cases, the temperature difference grows gradually along the length of the line, which is to be expected due to the flowwise accumulation of error. The maximum error at the end of the 5 section lumped parameter model called LINE5 is 3.7 degrees. Since the temperature in that lump changed by about 110 degrees (in a mere 15 seconds), this represents a maximum error of 3.4% for a model that was acknowledged to exceed good modeling practices. Each time the spatial resolution was increased by a factor of two (in the submodels LINE10 and LINE20), the error diminished by the same factor. With lumped parameter methods, the aforementioned rules–of–thumb that led to a breakdown into 10 sections yielded very acceptable accuracies.

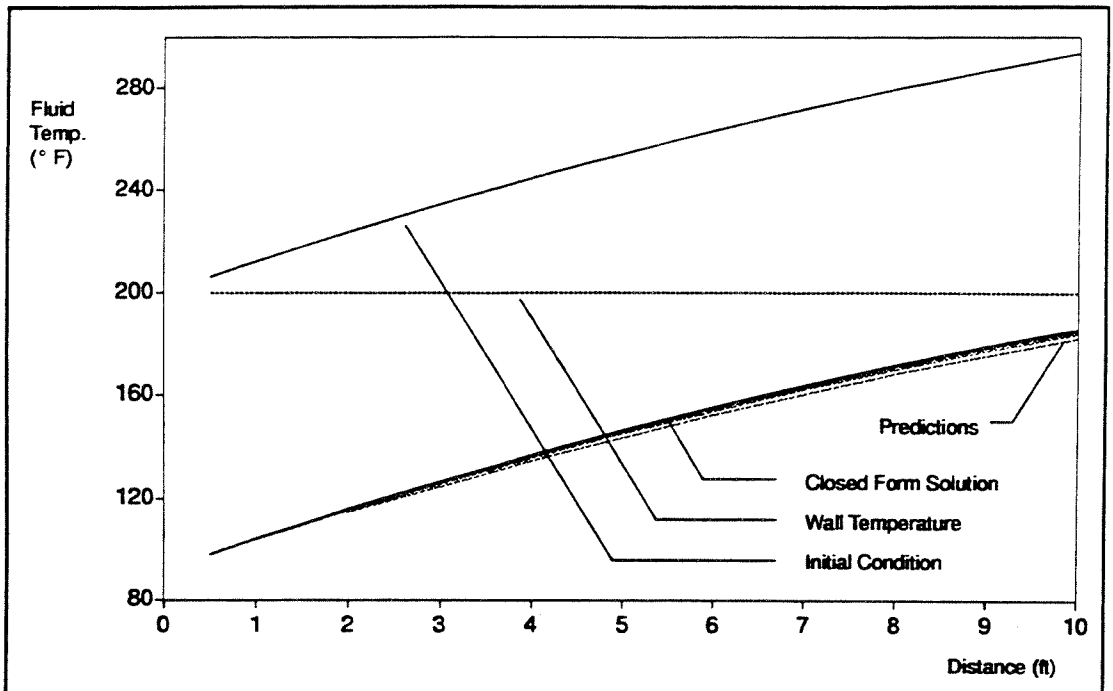


Figure H-2 Temperature Profiles for All Models

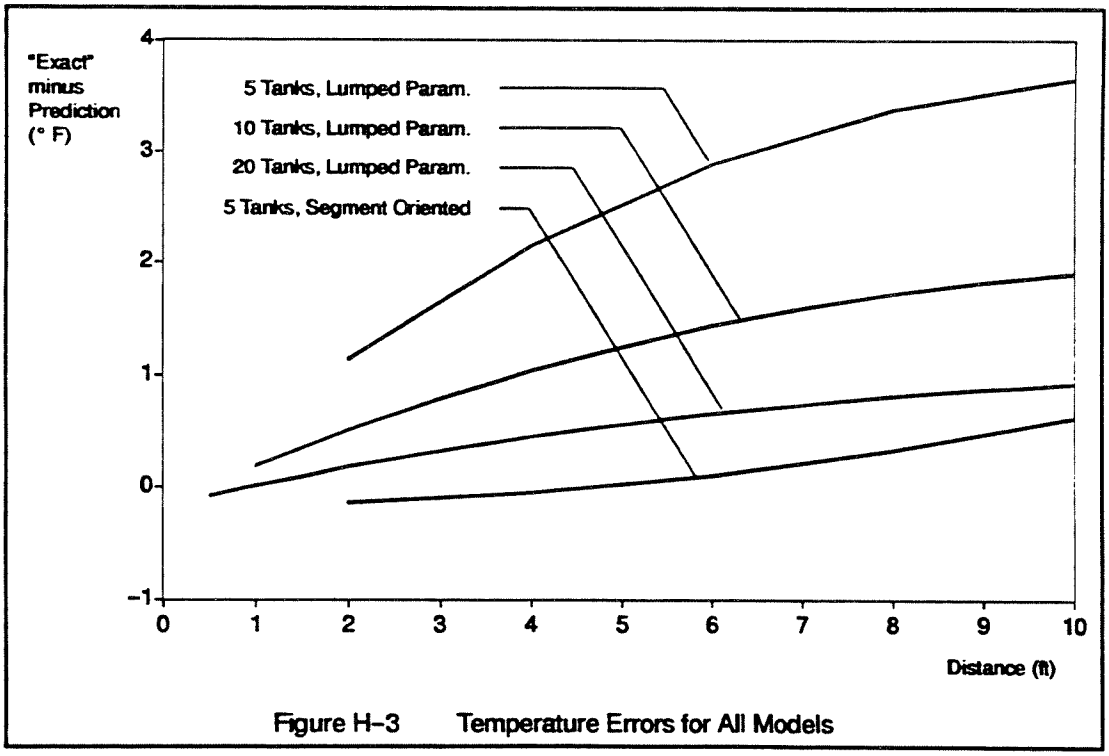


Figure H-3 Temperature Errors for All Models

Even the 20 tank lumped parameter model (LINE20) was outperformed by the 5 tank segment–oriented model (LINE5S), which achieved the best accuracy of all. Curiously, the accumulation of error in the lumped parameter models seems to have an asymptotic limit, whereas the error in the segment–oriented model, while very small, appears to grow without bound.

The success of the segment–oriented ties in this example should be viewed with some caution. Their forte is single–phase incompressible flows with spatially–invariant wall temperatures and relatively constant fluid properties, all of which exist in by this example. In fact, for two–phase flows, the methods revert to those of lumped parameter ties. In most other cases, other important gradients will likely be present in fluid properties, wall temperatures, or even heat transfer coefficients. Capturing such gradients requires adequate resolution. Still, for preliminary design of single–phase heat exchangers, segment–oriented ties represent a significant tool for model reduction.

Input File

```
HEADER OPTIONS DATA
TITLE WATER LINE, COMPARISON WITH ANALYTIC RESULT
         MODEL              = WLINE
         OUTPUT             = waterline.out
         USER1              = waterline.comp
         USER2              = waterline.terr
         NODEBUG
C
HEADER USER DATA,GLOBAL
         PI       = 3.141592654
         ALPHA    = 0.0
         BETA     = 0.0
C
HEADER NODE DATA,WALL
         -1,200.0,0.0                $ WALL BOUNDARY
C
HEADER OPERATIONS DATA
BUILDF CONF,LINE5,LINE5S,LINE10,LINE20
BUILD CONF,WALL
C
C PRINT OUT CP AND DENSITY, AND TIME/SPACE CONSTANTS
C
DEFMOD LINE20
         DEN                = VDL(TL1-ABSZRO,LINE20.FI)
         CP                 = VCPF(PL1-PATMOS,TL1-ABSZRO,LINE20.FI)
         WAREA              = 4.0*TLEN1*AF1/DH1
         HTC                = UA1/WAREA
         ALPHA              = 0.25*DH1*CP*DEN/HTC
         BETA               = SMFR100*CP/(HTC*PI*DH1)
         WRITE(NUSER1,100)CP,DEN,CP*DEN,ALPHA,BETA
100      FORMAT(1P,
     .  ' Cp       : ',G13.6,' BTU/LBm-F'/
     .  ' DEN      : ',G13.6,' LBm/FT3'/
     .  ' Cp*DEN   : ',G13.6,' BTU/FT3-F'/
     .  ' ALPHA    : ',G13.6,' HR'/
     .  ' BETA     : ',G13.6,' FT'/)
         WRITE(NUSER1,101)
         WRITE(NUSER2,102)
101      FORMAT(//' SEG NO.    DIST(FT)',9X,'TINIT',10X,'T EQN'
     .   ,10X,'T 20 ',10X,'T 10 ',10X,'T 5LP',10X,'T 5SG'/)
102      FORMAT(//' SEG NO.    DIST(FT)',9X,'E 20',10X,'E 10 '
     .   ,10X,'E 5LP',10X,'E 5SG'/)
C
         CALL FASTIC
C
```

```
C PRESSURES NOW STABLE: SET UP IC FOR TRANSIENT
DEFMOD WALL
        DO 5 I=1,5
                DIST            = 2.0 + 2.0*FLOAT(I-1)
                TEMP            = T1*(2.0-EXP(-DIST/BETA))
                CALL CHGLMP('LINE5',I,'TL',TEMP,'PL')
                CALL CHGLMP('LINE5S',I,'TL',TEMP,'PL')
5       CONTINUE
        DO 10 I=1,10
                DIST            = 1.0 + FLOAT(I-1)
                TEMP            = T1*(2.0-EXP(-DIST/BETA))
                CALL CHGLMP('LINE10',I,'TL',TEMP,'PL')
10      CONTINUE
        DO 20 I=1,20
                DIST            = 0.5 + 0.5*FLOAT(I-1)
                TEMP            = T1*(2.0-EXP(-DIST/BETA))
                CALL CHGLMP('LINE20',I,'TL',TEMP,'PL')
20      CONTINUE
C
        CALL FORWRD
C
C FIND EXACT SOLUTION FOR COMPARISON
C
        DO 201 I=1,20
                DIST  = 0.5 + 0.5*FLOAT(I-1)
                TEMP  = T1*(1.0-EXP(-DIST/BETA) + EXP(-TIMEN/ALPHA))
                TINIT = T1*(2.0-EXP(-DIST/BETA))
                TEMP1 = 0.
                TEMP2 = 0.
                TEMP3 = 0.
                IF(MOD(I,2) .EQ. 0)THEN
                        II              = I/2
                        TEMP1           = LINE10.TL(1+II-1)
                        IF(MOD(I,4) .EQ. 0) THEN
                                II      = I/4
                                TEMP2   = LINE5.TL(1+II-1)
                                TEMP3   = LINE5S.TL(1+II-1)
                        ENDIF
                ENDIF
                WRITE(NUSER1,200)I,DIST,TINIT,
                        TEMP,LINE20.TL(1+I-1),TEMP1,TEMP2,TEMP3
                IF(TEMP1 .EQ. 0.0) TEMP1 = TEMP
                IF(TEMP2 .EQ. 0.0) TEMP2 = TEMP
                IF(TEMP3 .EQ. 0.0) TEMP3 = TEMP
                WRITE(NUSER2,200)I,DIST,TEMP-LINE20.TL(1+I-1),
                 TEMP-TEMP1,TEMP-TEMP2,TEMP-TEMP3
F200            FORMAT(1X,I3,5X,1P,7G15.6)
201     CONTINUE
```

```
HEADER CONTROL DATA,GLOBAL
         NLOOPS            = 50
         UID               = ENG
C MAKE SURE TIME STEP COVERS CHANGING INLET PROFILE
         DTMAXF            = 1.0E-5
C 12.25 SEC IS TIME FOR FLAT-FRONT TO MOVE ALL THE WAY THROUGH
         TIMEND            = 15.0/3600.0
         OUTPTF            = 1.0
         OUTPUT            = 1.0
C
HEADER FLOW DATA,LINE5,FID=9718
C
PA DEF, FR        = 1000.0
LU DEF, TL        = 200.0, PL       = 100.,  XL = 0.0
LU PLEN,100                                  $ UPSTREAM PLENUM
PA CONN,100,5,100, DEV = MFRSET              $ SET FLOWRATE
M LINE,1,D,1,1,100,                  NSEG    = 5
         TLENT    = 10.0,            DHS     = 1.0/12.0
         PA       = STUBE, UPF       = 0.0
M GENT,2,HTU,1,1,WALL.1,N=5,
         UA       = 242.0*3.141592654/12.0   * 2.0
         NINC     = 0
C
HEADER FLOW DATA,LINE10,FID=9718
PA DEF, FR        = 1000.0
LU DEF, TL        = 200.0, PL       = 100.,  XL = 0.0
LU PLEN,100                                  $ UPSTREAM PLENUM
PA CONN,100,10,100, DEV = MFRSET    $ SET FLOWRATE
M LINE,1,D,1,1,100,                  NSEG    = 10
         TLENT    = 10.0,            DHS     = 1.0/12.0
         PA       = STUBE, UPF       = 0.0
M GENT,2,HTU,1,1,WALL.1,N=10,
         UA       = 242.0*3.141592654/12.0
         NINC     = 0
C
HEADER FLOW DATA,LINE20,FID=9718
PA DEF, FR        = 1000.0
LU DEF, TL        = 200.0, PL       = 100.,  XL = 0.0
LU PLEN,100                                  $ UPSTREAM PLENUM
PA CONN,100,20,100, DEV = MFRSET    $ SET FLOWRATE
M LINE,1,D,1,1,100,                  NSEG    = 20
         TLENT    = 10.0,            DHS     = 1.0/12.0
         PA       = STUBE, UPF       = 0.0
M GENT,2,HTU,1,1,WALL.1,N=20,
         UA       = 242.0*3.141592654/12.0   * 0.5
         NINC     = 0
C
```

```
HEADER FLOW DATA,LINE5S,FID=9718
C
PA DEF, FR          = 1000.0
LU DEF, TL          = 200.0, PL         = 100.,   XL = 0.0
LU PLEN,100                                       $ UPSTREAM PLENUM
PA CONN,100,5,100, DEV = MFRSET                   $ SET FLOWRATE
M LINE,1,D,1,1,100,                     NSEG      = 5
          TLENT     = 10.0,             DHS       = 1.0/12.0
          PA        = STUBE, UPF        = 0.0
M GENT,2,HTUS,1,1,WALL.1,N=5
          UA        = 242.0*3.141592654/12.0    * 2.0
          NINC      = 0
C
HEADER OUTPUT CALLS, LINE10
          CALL LMPTAB('ALL')
          CALL TIETAB('ALL')
C
HEADER FLOGIC0, LINE10
          TEMP      = WALL.T1*EXP(-TIMEN/ALPHA)
          CALL CHGLMP('LINE5',100,'TL',TEMP,'PL')
          CALL CHGLMP('LINE5S',100,'TL',TEMP,'PL')
          CALL CHGLMP('LINE10',100,'TL',TEMP,'PL')
          CALL CHGLMP('LINE20',100,'TL',TEMP,'PL')
C
HEADER FPROP DATA,9718,ENG,-460.0
C SIMPLIFIED WATER
          V         = 2.32
          K         = 0.350
          CP        = 1.0
          D         = 62.4
END OF DATA
```

Processor Output

CP	:	1.00000	BTU/LBM-F
DEN	:	62.4000	LBM/FT3
CP*DEN	:	62.4000	BTU/FT3-F
ALPHA	:	5.371902E-03	HR
BETA	:	15.7840	FT

SEG NO.	DIST(FT)	TINIT	T EQN	T 20	T 10	T 5LP	T 5SG
1	0.500000	206.236	98.3180	98.3894	0.	0.	0.
2	1.00000	212.278	104.360	104.341	104.163	0.	0.
3	1.50000	218.131	110.213	110.108	0.	0.	0.
4	2.00000	223.802	115.884	115.698	115.365	114.735	116.011
5	2.50000	229.296	121.378	121.116	0.	0.	0.
6	3.00000	234.619	126.701	126.368	125.898	0.	0.
7	3.50000	239.776	131.858	131.458	0.	0.	0.
8	4.00000	244.772	136.854	136.392	135.803	134.695	136.893
9	4.50000	249.612	141.694	141.175	0.	0.	0.
10	5.00000	254.301	146.383	145.810	145.119	0.	0.
11	5.50000	258.845	150.926	150.303	0.	0.	0.
12	6.00000	263.246	155.328	154.658	153.879	152.427	155.217
13	6.50000	267.510	159.592	158.879	0.	0.	0.
14	7.00000	271.641	163.723	162.970	162.119	0.	0.
15	7.50000	275.644	167.725	166.935	0.	0.	0.
16	8.00000	279.521	171.603	170.778	169.870	168.204	171.263
17	8.50000	283.278	175.360	174.503	0.	0.	0.
18	9.00000	286.917	178.999	178.112	177.166	0.	0.
19	9.50000	290.444	182.525	181.611	0.	0.	0.
20	10.0000	293.860	185.941	185.003	184.035	182.267	185.311

SEG NO.	DIST(FT)	E 20	E 10	E 5LP	E 5SG
1	0.500000	-7.140350E-02	0.	0.	0.
2	1.00000	1.843262E-02	0.196899	0.	0.
3	1.50000	0.104752	0.	0.	0.
4	2.00000	0.185905	0.519218	1.14867	-0.126900
5	2.50000	0.261765	0.	0.	0.
6	3.00000	0.332802	0.802834	0.	0.
7	3.50000	0.399307	0.	0.	0.
8	4.00000	0.461334	1.05026	2.15891	-3.890991E-02
9	4.50000	0.519318	0.	0.	0.
10	5.00000	0.573227	1.26457	0.	0.
11	5.50000	0.623367	0.	0.	0.
12	6.00000	0.669922	1.44873	2.90033	0.110535
13	6.50000	0.713135	0.	0.	0.
14	7.00000	0.753281	1.60435	0.	0.
15	7.50000	0.790451	0.	0.	0.
16	8.00000	0.824951	1.73248	3.39844	0.339478
17	8.50000	0.857086	0.	0.	0.
18	9.00000	0.886765	1.83318	0.	0.
19	9.50000	0.914108	0.	0.	0.
20	10.0000	0.938599	1.90613	3.67468	0.630005

FLUID SUBMODEL NAME = LINE5 ; FLUID NO. = 9718

LOOPCT = 12
CONVERGENCE STATUS = SUBMODEL CONVERGED AS OF 10 ITERATIONS

LUMP PARAMETER TABULATION FOR SUBMODEL LINE5

LUMP	TYPE	TEMP	PRESSURE	QUALITY	DENSITY	ENTHALPY	HEAT RATE	MASS RATE	ENERGY RATE
1	TANK	200.0	100.0	0.	62.40	660.0	-7.7338E-03	0.	-7.7338E-03
2	TANK	200.0	99.99	0.	62.40	660.0	-7.7338E-03	0.	-7.7338E-03
3	TANK	200.0	99.99	0.	62.40	660.0	-7.7338E-03	0.	-7.7338E-03
4	TANK	200.0	99.98	0.	62.40	660.0	-7.7338E-03	0.	-7.7338E-03
5	TANK	200.0	99.98	0.	62.40	660.0	-7.7338E-03	0.	-7.7338E-03
100	PLEN	200.0	100.0	0.	62.40	660.0	0.	0.	0.

FLUID SUBMODEL NAME = LINE5S ; FLUID NO. = 9718

LOOPCT = 12
CONVERGENCE STATUS = SUBMODEL CONVERGED AS OF 10 ITERATIONS

LUMP PARAMETER TABULATION FOR SUBMODEL LINE5S

LUMP	TYPE	TEMP	PRESSURE	QUALITY	DENSITY	ENTHALPY	HEAT RATE	MASS RATE	ENERGY RATE
1	TANK	200.0	100.0	0.	62.40	660.0	-3.8669E-03	0.	-3.8669E-03
2	TANK	200.0	99.99	0.	62.40	660.0	-7.7338E-03	0.	-7.7338E-03
3	TANK	200.0	99.99	0.	62.40	660.0	-7.7338E-03	0.	-7.7338E-03
4	TANK	200.0	99.98	0.	62.40	660.0	-7.7338E-03	0.	-7.7338E-03
5	TANK	200.0	99.98	0.	62.40	660.0	-7.7338E-03	0.	-7.7338E-03
100	PLEN	200.0	100.0	0.	62.40	660.0	0.	0.	0.

MODEL = WLINE
FASTIC

WATER LINE, COMPARISON WITH ANALYTIC RESULT

FLUID SUBMODEL NAME = LINE10 ; FLUID NO. = 9718

LOOPCT = 12
CONVERGENCE STATUS = SUBMODEL CONVERGED AS OF 10 ITERATIONS

LUMP PARAMETER TABULATION FOR SUBMODEL LINE10

LUMP	TYPE	TEMP	PRESSURE	QUALITY	DENSITY	ENTHALPY	HEAT RATE	MASS RATE	ENERGY RATE
1	TANK	200.0	100.0	0.	62.40	660.0	-3.8669E-03	0.	-3.8669E-03
2	TANK	200.0	100.0	0.	62.40	660.0	-3.8669E-03	0.	-3.8669E-03
3	TANK	200.0	99.99	0.	62.40	660.0	-3.8669E-03	0.	-3.8669E-03
4	TANK	200.0	99.99	0.	62.40	660.0	-3.8669E-03	0.	-3.8669E-03
5	TANK	200.0	99.99	0.	62.40	660.0	-3.8669E-03	0.	-3.8669E-03
6	TANK	200.0	99.99	0.	62.40	660.0	-3.8669E-03	0.	-3.8669E-03
7	TANK	200.0	99.99	0.	62.40	660.0	-3.8669E-03	0.	-3.8669E-03
8	TANK	200.0	99.98	0.	62.40	660.0	-3.8669E-03	0.	-3.8669E-03
9	TANK	200.0	99.98	0.	62.40	660.0	-3.8669E-03	0.	-3.8669E-03
10	TANK	200.0	99.98	0.	62.40	660.0	-3.8669E-03	0.	-3.8669E-03
100	PLEN	200.0	100.0	0.	62.40	660.0	0.	0.	0.

MODEL = WLINE
FASTIC

WATER LINE, COMPARISON WITH ANALYTIC RESULT

FLUID SUBMODEL NAME = LINE20 ; FLUID NO. = 9718

LOOPCT = 12
CONVERGENCE STATUS = SUBMODEL CONVERGED AS OF 11 ITERATIONS

LUMP PARAMETER TABULATION FOR SUBMODEL LINE20

LUMP	TYPE	TEMP	PRESSURE	QUALITY	DENSITY	ENTHALPY	HEAT RATE	MASS RATE	ENERGY RATE
1	TANK	200.0	100.0	0.	62.40	660.0	-1.9335E-03	0.	-1.9335E-03
2	TANK	200.0	100.0	0.	62.40	660.0	-1.9335E-03	0.	-1.9335E-03
3	TANK	200.0	100.0	0.	62.40	660.0	-1.9335E-03	0.	-1.9335E-03
4	TANK	200.0	100.0	0.	62.40	660.0	-1.9335E-03	0.	-1.9335E-03
5	TANK	200.0	100.0	0.	62.40	660.0	-1.9335E-03	0.	-1.9335E-03
6	TANK	200.0	99.99	0.	62.40	660.0	-1.9335E-03	0.	-1.9335E-03
7	TANK	200.0	99.99	0.	62.40	660.0	-1.9335E-03	0.	-1.9335E-03
8	TANK	200.0	99.99	0.	62.40	660.0	-1.9335E-03	0.	-1.9335E-03
9	TANK	200.0	99.99	0.	62.40	660.0	-1.9335E-03	0.	-1.9335E-03
10	TANK	200.0	99.99	0.	62.40	660.0	-1.9335E-03	0.	-1.9335E-03
11	TANK	200.0	99.99	0.	62.40	660.0	-1.9335E-03	0.	-1.9335E-03
12	TANK	200.0	99.99	0.	62.40	660.0	-1.9335E-03	0.	-1.9335E-03
13	TANK	200.0	99.99	0.	62.40	660.0	-1.9335E-03	0.	-1.9335E-03
14	TANK	200.0	99.99	0.	62.40	660.0	-1.9335E-03	0.	-1.9335E-03
15	TANK	200.0	99.99	0.	62.40	660.0	-1.9335E-03	0.	-1.9335E-03
16	TANK	200.0	99.98	0.	62.40	660.0	-1.9335E-03	0.	-1.9335E-03
17	TANK	200.0	99.98	0.	62.40	660.0	-1.9335E-03	0.	-1.9335E-03
18	TANK	200.0	99.98	0.	62.40	660.0	-1.9335E-03	0.	-1.9335E-03
19	TANK	200.0	99.98	0.	62.40	660.0	-1.9335E-03	0.	-1.9335E-03
20	TANK	200.0	99.98	0.	62.40	660.0	-1.9335E-03	0.	-1.9335E-03
100	PLEN	200.0	100.0	0.	62.40	660.0	0.	0.	0.

MODEL = WLINE
FASTIC

WATER LINE, COMPARISON WITH ANALYTIC RESULT

FLUID SUBMODEL NAME = LINE5 ; FLUID NO. = 9718

LOOPCT = 12
CONVERGENCE STATUS = SUBMODEL CONVERGED AS OF 10 ITERATIONS

TIE PARAMETER TABULATION FOR SUBMODEL LINE5

TIE	TYPE	UA	QTIE	LUMP	TEMP.	NODE	TEMP.	PATH 1	FRACT	PATH 2	FRACT
1	BTU	126.7	-7.7338E-03	1	200.0	WALL.1	200.0				
2	BTU	126.7	-7.7338E-03	2	200.0	WALL.1	200.0				
3	BTU	126.7	-7.7338E-03	3	200.0	WALL.1	200.0				
4	BTU	126.7	-7.7338E-03	4	200.0	WALL.1	200.0				
5	BTU	126.7	-7.7338E-03	5	200.0	WALL.1	200.0				

MODEL = WLINE
FASTIC

WATER LINE, COMPARISON WITH ANALYTIC RESULT

FLUID SUBMODEL NAME = LINE5S ; FLUID NO. = 9718

LOOPCT = 12
CONVERGENCE STATUS = SUBMODEL CONVERGED AS OF 10 ITERATIONS

TIE PARAMETER TABULATION FOR SUBMODEL LINE5S

TIE	TYPE	UA	QTIE	LUMP	TEMP.	NODE	TEMP.	PATH 1	FRACT	PATH 2	FRACT
1	BTUS	126.7	-3.8669E-03	1	200.0	WALL.1	200.0				
2	BTUS	126.7	-7.7338E-03	2	200.0	WALL.1	200.0				
3	BTUS	126.7	-7.7338E-03	3	200.0	WALL.1	200.0				
4	BTUS	126.7	-7.7338E-03	4	200.0	WALL.1	200.0				
5	BTUS	126.7	-7.7338E-03	5	200.0	WALL.1	200.0				

MODEL = WLINE
FASTIC

WATER LINE, COMPARISON WITH ANALYTIC RESULT

FLUID SUBMODEL NAME = LINE10 ; FLUID NO. = 9718

LOOPCT = 12
CONVERGENCE STATUS = SUBMODEL CONVERGED AS OF 10 ITERATIONS

TIE PARAMETER TABULATION FOR SUBMODEL LINE10

TIE	TYPE	UA	QTIE	LUMP	TEMP.	NODE	TEMP.	PATH 1	FRACT	PATH 2	FRACT
1	BTU	63.36	-3.8669E-03	1	200.0	WALL.1	200.0				
2	BTU	63.36	-3.8669E-03	2	200.0	WALL.1	200.0				
3	BTU	63.36	-3.8669E-03	3	200.0	WALL.1	200.0				
4	BTU	63.36	-3.8669E-03	4	200.0	WALL.1	200.0				
5	BTU	63.36	-3.8669E-03	5	200.0	WALL.1	200.0				
6	BTU	63.36	-3.8669E-03	6	200.0	WALL.1	200.0				
7	BTU	63.36	-3.8669E-03	7	200.0	WALL.1	200.0				
8	BTU	63.36	-3.8669E-03	8	200.0	WALL.1	200.0				
9	BTU	63.36	-3.8669E-03	9	200.0	WALL.1	200.0				
10	BTU	63.36	-3.8669E-03	10	200.0	WALL.1	200.0				

MODEL = WLINE
FASTIC

WATER LINE, COMPARISON WITH ANALYTIC RESULT

FLUID SUBMODEL NAME = LINE20 ; FLUID NO. = 9718

LOOPCT = 12
CONVERGENCE STATUS = SUBMODEL CONVERGED AS OF 11 ITERATIONS

TIE PARAMETER TABULATION FOR SUBMODEL LINE20

TIE	TYPE	UA	QTIE	LUMP	TEMP.	NODE	TEMP.	PATH 1	FRACT	PATH 2	FRACT
1	BTU	31.68	-1.9335E-03	1	200.0	WALL.1	200.0				
2	BTU	31.68	-1.9335E-03	2	200.0	WALL.1	200.0				
3	BTU	31.68	-1.9335E-03	3	200.0	WALL.1	200.0				
4	BTU	31.68	-1.9335E-03	4	200.0	WALL.1	200.0				
5	BTU	31.68	-1.9335E-03	5	200.0	WALL.1	200.0				
6	BTU	31.68	-1.9335E-03	6	200.0	WALL.1	200.0				
7	BTU	31.68	-1.9335E-03	7	200.0	WALL.1	200.0				
8	BTU	31.68	-1.9335E-03	8	200.0	WALL.1	200.0				
9	BTU	31.68	-1.9335E-03	9	200.0	WALL.1	200.0				
10	BTU	31.68	-1.9335E-03	10	200.0	WALL.1	200.0				
11	BTU	31.68	-1.9335E-03	11	200.0	WALL.1	200.0				
12	BTU	31.68	-1.9335E-03	12	200.0	WALL.1	200.0				
13	BTU	31.68	-1.9335E-03	13	200.0	WALL.1	200.0				
14	BTU	31.68	-1.9335E-03	14	200.0	WALL.1	200.0				
15	BTU	31.68	-1.9335E-03	15	200.0	WALL.1	200.0				
16	BTU	31.68	-1.9335E-03	16	200.0	WALL.1	200.0				
17	BTU	31.68	-1.9335E-03	17	200.0	WALL.1	200.0				
18	BTU	31.68	-1.9335E-03	18	200.0	WALL.1	200.0				
19	BTU	31.68	-1.9335E-03	19	200.0	WALL.1	200.0				
20	BTU	31.68	-1.9335E-03	20	200.0	WALL.1	200.0				

MODEL = WLINE
FORWRD

WATER LINE, COMPARISON WITH ANALYTIC RESULT

FLUID SUBMODEL NAME = LINE5 ; FLUID NO. = 9718

MAX TIME STEP = 3.337795E-04 ; LIMITING TANK = 1 REASON = TEMPERATURE CHANGE LIMIT
LAST TIME STEP = 6.028451E-06 VS. DTMAXF/DTMINF = 1.000000E-05 / 0. ; AVERAGE TIME STEP = 9.602842E-06
PROBLEM TIME TIMEN = 4.166667E-03 VS. TIMEND = 4.166667E-03

LUMP PARAMETER TABULATION FOR SUBMODEL LINE5

LUMP	TYPE	TEMP	PRESSURE	QUALITY	DENSITY	ENTHALPY	HEAT RATE	MASS RATE	ENERGY RATE
1	TANK	114.7	100.0	0.	62.40	574.7	1.0791E+04	0.	-1.1759E+04
2	TANK	134.7	99.99	0.	62.40	594.7	8262.	0.	-1.1698E+04
3	TANK	152.4	99.99	0.	62.40	612.4	6015.	0.	-1.1718E+04
4	TANK	168.2	99.98	0.	62.40	628.2	4016.	0.	-1.1762E+04
5	TANK	182.3	99.98	0.	62.40	642.2	2234.	0.	-1.1828E+04
100	PLEN	92.19	100.0	0.	62.40	552.2	0.	0.	9.0081E+04

MODEL = WLINE
FORWRD

WATER LINE, COMPARISON WITH ANALYTIC RESULT

FLUID SUBMODEL NAME = LINE5S ; FLUID NO. = 9718

MAX TIME STEP = 3.351145E-04 ; LIMITING TANK = 1 REASON = TEMPERATURE CHANGE LIMIT
LAST TIME STEP = 6.028451E-06 VS. DTMAXF/DTMINF = 1.000000E-05 / 0. ; AVERAGE TIME STEP = 9.602842E-06
PROBLEM TIME TIMEN = 4.166667E-03 VS. TIMEND = 4.166667E-03

LUMP PARAMETER TABULATION FOR SUBMODEL LINE5S

LUMP	TYPE	TEMP	PRESSURE	QUALITY	DENSITY	ENTHALPY	HEAT RATE	MASS RATE	ENERGY RATE
1	TANK	116.0	100.0	0.	62.40	576.0	1.2082E+04	0.	-1.1744E+04
2	TANK	136.9	99.99	0.	62.40	596.9	9243.	0.	-1.1638E+04
3	TANK	155.2	99.99	0.	62.40	615.2	6756.	0.	-1.1568E+04
4	TANK	171.3	99.98	0.	62.40	631.2	4570.	0.	-1.1477E+04
5	TANK	185.3	99.98	0.	62.40	645.3	2639.	0.	-1.1409E+04
100	PLEN	92.19	100.0	0.	62.40	552.2	0.	0.	9.3126E+04

MODEL = WLINE
FORWRD

WATER LINE, COMPARISON WITH ANALYTIC RESULT

FLUID SUBMODEL NAME = LINE10 ; FLUID NO. = 9718

MAX TIME STEP = 3.276301E-04 ; LIMITING TANK = 1 REASON = TEMPERATURE CHANGE LIMIT
LAST TIME STEP = 6.028451E-06 VS. DTMAXF/DTMINF = 1.000000E-05 / 0. ; AVERAGE TIME STEP = 9.602842E-06
PROBLEM TIME TIMEN = 4.166667E-03 VS. TIMEND = 4.166667E-03

LUMP PARAMETER TABULATION FOR SUBMODEL LINE10

LUMP	TYPE	TEMP	PRESSURE	QUALITY	DENSITY	ENTHALPY	HEAT RATE	MASS RATE	ENERGY RATE
1	TANK	104.2	100.0	0.	62.40	564.1	6065.	0.	-5912.
2	TANK	115.4	100.0	0.	62.40	575.3	5356.	0.	-5846.
3	TANK	125.9	99.99	0.	62.40	585.9	4688.	0.	-5845.
4	TANK	135.8	99.99	0.	62.40	595.8	4061.	0.	-5845.
5	TANK	145.1	99.99	0.	62.40	605.1	3470.	0.	-5845.
6	TANK	153.9	99.99	0.	62.40	613.8	2915.	0.	-5845.
7	TANK	162.1	99.99	0.	62.40	622.1	2393.	0.	-5846.
8	TANK	169.9	99.98	0.	62.40	629.8	1902.	0.	-5850.
9	TANK	177.2	99.98	0.	62.40	637.1	1440.	0.	-5855.
10	TANK	184.0	99.98	0.	62.40	644.0	1005.	0.	-5864.
100	PLEN	92.19	100.0	0.	62.40	552.2	0.	0.	9.1850E+04

MODEL = WLINE
FORWRD

WATER LINE, COMPARISON WITH ANALYTIC RESULT

FLUID SUBMODEL NAME = LINE20 ; FLUID NO. = 9718

MAX TIME STEP = 3.242624E-04 ; LIMITING TANK = 1 REASON = TEMPERATURE CHANGE LIMIT
LAST TIME STEP = 6.028451E-06 VS. DTMAXF/DTMINF = 1.000000E-05 / 0. ; AVERAGE TIME STEP = 9.602842E-06
PROBLEM TIME TIMEN = 4.166667E-03 VS. TIMEND = 4.166667E-03

LUMP PARAMETER TABULATION FOR SUBMODEL LINE20

LUMP	TYPE	TEMP	PRESSURE	QUALITY	DENSITY	ENTHALPY	HEAT RATE	MASS RATE	ENERGY RATE
1	TANK	98.39	100.0	0.	62.40	558.4	3215.	0.	-2989.
2	TANK	104.3	100.0	0.	62.40	564.3	3027.	0.	-2925.
3	TANK	110.1	100.0	0.	62.40	570.1	2844.	0.	-2923.
4	TANK	115.7	100.0	0.	62.40	575.7	2667.	0.	-2923.
5	TANK	121.1	100.0	0.	62.40	581.1	2496.	0.	-2923.
6	TANK	126.4	99.99	0.	62.40	586.3	2329.	0.	-2923.
7	TANK	131.5	99.99	0.	62.40	591.4	2168.	0.	-2922.
8	TANK	136.4	99.99	0.	62.40	596.4	2012.	0.	-2922.
9	TANK	141.2	99.99	0.	62.40	601.1	1860.	0.	-2922.
10	TANK	145.8	99.99	0.	62.40	605.8	1713.	0.	-2922.
11	TANK	150.3	99.99	0.	62.40	610.3	1571.	0.	-2922.
12	TANK	154.7	99.99	0.	62.40	614.6	1433.	0.	-2922.
13	TANK	158.9	99.99	0.	62.40	618.8	1299.	0.	-2922.
14	TANK	163.0	99.99	0.	62.40	622.9	1170.	0.	-2921.
15	TANK	166.9	99.99	0.	62.40	626.9	1044.	0.	-2921.
16	TANK	170.8	99.98	0.	62.40	630.7	922.4	0.	-2921.
17	TANK	174.5	99.98	0.	62.40	634.5	804.4	0.	-2920.
18	TANK	178.1	99.98	0.	62.40	638.1	690.1	0.	-2920.
19	TANK	181.6	99.98	0.	62.40	641.6	579.2	0.	-2919.
20	TANK	185.0	99.98	0.	62.40	645.0	471.8	0.	-2920.
100	PLEN	92.19	100.0	0.	62.40	552.2	0.	0.	9.2818E+04

MODEL = WLINE
FORWRD

WATER LINE, COMPARISON WITH ANALYTIC RESULT

FLUID SUBMODEL NAME = LINE5 ; FLUID NO. = 9718

MAX TIME STEP = 3.337795E-04 ; LIMITING TANK = 1 REASON = TEMPERATURE CHANGE LIMIT
LAST TIME STEP = 6.028451E-06 VS. DTMAXF/DTMINF = 1.000000E-05 / 0. ; AVERAGE TIME STEP = 9.602842E-06
PROBLEM TIME TIMEN = 4.166667E-03 VS. TIMEND = 4.166667E-03

TIE PARAMETER TABULATION FOR SUBMODEL LINE5

TIE	TYPE	UA	QTIE	LUMP	TEMP.	NODE	TEMP.	PATH 1	FRACT	PATH 2	FRACT
1	BTU	126.7	1.0791E+04	1	114.7	WALL.1	200.0				
2	BTU	126.7	8262.	2	134.7	WALL.1	200.0				
3	BTU	126.7	6015.	3	152.4	WALL.1	200.0				
4	BTU	126.7	4016.	4	168.2	WALL.1	200.0				
5	BTU	126.7	2234.	5	182.3	WALL.1	200.0				

MODEL = WLINE
FORWRD

WATER LINE, COMPARISON WITH ANALYTIC RESULT

FLUID SUBMODEL NAME = LINE5S ; FLUID NO. = 9718

MAX TIME STEP = 3.351145E-04 ; LIMITING TANK = 1 REASON = TEMPERATURE CHANGE LIMIT
LAST TIME STEP = 6.028451E-06 VS. DTMAXF/DTMINF = 1.000000E-05 / 0. ; AVERAGE TIME STEP = 9.602842E-06
PROBLEM TIME TIMEN = 4.166667E-03 VS. TIMEND = 4.166667E-03

TIE PARAMETER TABULATION FOR SUBMODEL LINE5S

TIE	TYPE	UA	QTIE	LUMP	TEMP.	NODE	TEMP.	PATH 1	FRACT	PATH 2	FRACT
1	BTUS	126.7	1.2082E+04	1	116.0	WALL.1	200.0				
2	BTUS	126.7	9243.	2	136.9	WALL.1	200.0				
3	BTUS	126.7	6756.	3	155.2	WALL.1	200.0				
4	BTUS	126.7	4570.	4	171.3	WALL.1	200.0				
5	BTUS	126.7	2639.	5	185.3	WALL.1	200.0				

MODEL = WLINE
FORWRD

WATER LINE, COMPARISON WITH ANALYTIC RESULT

FLUID SUBMODEL NAME = LINE10 ; FLUID NO. = 9718

MAX TIME STEP = 3.276301E-04 ; LIMITING TANK = 1 REASON = TEMPERATURE CHANGE LIMIT
LAST TIME STEP = 6.028451E-06 VS. DTMAXF/DTMINF = 1.000000E-05 / 0. ; AVERAGE TIME STEP = 9.602842E-06
PROBLEM TIME TIMEN = 4.166667E-03 VS. TIMEND = 4.166667E-03

TIE PARAMETER TABULATION FOR SUBMODEL LINE10

TIE	TYPE	UA	QTIE	LUMP	TEMP.	NODE	TEMP.	PATH 1	FRACT	PATH 2	FRACT
1	BTU	63.36	6065.	1	104.2	WALL.1	200.0				
2	BTU	63.36	5356.	2	115.4	WALL.1	200.0				
3	BTU	63.36	4688.	3	125.9	WALL.1	200.0				
4	BTU	63.36	4061.	4	135.8	WALL.1	200.0				
5	BTU	63.36	3470.	5	145.1	WALL.1	200.0				
6	BTU	63.36	2915.	6	153.9	WALL.1	200.0				
7	BTU	63.36	2393.	7	162.1	WALL.1	200.0				
8	BTU	63.36	1902.	8	169.9	WALL.1	200.0				
9	BTU	63.36	1440.	9	177.2	WALL.1	200.0				
10	BTU	63.36	1005.	10	184.0	WALL.1	200.0				

MODEL = WLINE
FORWRD

WATER LINE, COMPARISON WITH ANALYTIC RESULT

FLUID SUBMODEL NAME = LINE20 ; FLUID NO. = 9718

MAX TIME STEP = 3.242624E-04 ; LIMITING TANK = 1 REASON = TEMPERATURE CHANGE LIMIT
LAST TIME STEP = 6.028451E-06 VS. DTMAXF/DTMINF = 1.000000E-05 / 0. ; AVERAGE TIME STEP = 9.602842E-06
PROBLEM TIME TIMEN = 4.166667E-03 VS. TIMEND = 4.166667E-03

TIE PARAMETER TABULATION FOR SUBMODEL LINE20

TIE	TYPE	UA	QTIE	LUMP	TEMP.	NODE	TEMP.	PATH 1	FRACT	PATH 2	FRACT
1	BTU	31.68	3215.	1	98.39	WALL.1	200.0				
2	BTU	31.68	3027.	2	104.3	WALL.1	200.0				
3	BTU	31.68	2844.	3	110.1	WALL.1	200.0				
4	BTU	31.68	2667.	4	115.7	WALL.1	200.0				
5	BTU	31.68	2496.	5	121.1	WALL.1	200.0				
6	BTU	31.68	2329.	6	126.4	WALL.1	200.0				
7	BTU	31.68	2168.	7	131.5	WALL.1	200.0				
8	BTU	31.68	2012.	8	136.4	WALL.1	200.0				
9	BTU	31.68	1860.	9	141.2	WALL.1	200.0				
10	BTU	31.68	1713.	10	145.8	WALL.1	200.0				
11	BTU	31.68	1571.	11	150.3	WALL.1	200.0				
12	BTU	31.68	1433.	12	154.7	WALL.1	200.0				
13	BTU	31.68	1299.	13	158.9	WALL.1	200.0				
14	BTU	31.68	1170.	14	163.0	WALL.1	200.0				
15	BTU	31.68	1044.	15	166.9	WALL.1	200.0				
16	BTU	31.68	922.4	16	170.8	WALL.1	200.0				
17	BTU	31.68	804.4	17	174.5	WALL.1	200.0				
18	BTU	31.68	690.1	18	178.1	WALL.1	200.0				
19	BTU	31.68	579.2	19	181.6	WALL.1	200.0				
20	BTU	31.68	471.8	20	185.0	WALL.1	200.0				