Everything you've always wanted to know about RPN

but were afraid to pursue

COMPREHENSIVE MANUAL
FOR SCIENTIFIC CALCULATORS

CORVUS 500
APS MARK 55
OMRON 12-SR
and others

Published by T.K. Enterprises

$750

$3.00

Everything you've always wanted to know about RPN

but were afraid to pursue

COMPREHENSIVE MANUAL FOR
SCIENTIFIC CALCULATORS

Published by

tkenterprises
BETTER BUSINESS PRODUCTS

16611 Hawthorne Blvd., Lawndale, CA. 90260

TABLE OF CONTENTS

INTRODUCTION

ABOUT THE BOOK

The primary purpose of this book is to help you get the most out of your scientific calculator. In particular, we have oriented this book to the use of the CORVUS 500 because of the large number of functions it offers and because it makes use of Reverse Polish Notation (RPN). RPN is the most efficient means for expressing complex calculations – an entire section of this book is devoted to a discussion of this important technique.

This book is divided into two main parts. The first part describes the basic operation of the CORVUS 500. This description also applies to the APF Mark 55 and the OMRON 12SR calculators which are functionally identical to the CORVUS. Furthermore, the discussion of RPN is generally applicable and should provide the reader with a thorough understanding of RPN and an appreciation for its simplicity.

The second part of this book presents a selection of application problems and their accompanying solution programs. These sample problems are organized by application area. These areas include financial, statistical and simple algebraic calculations.

Every effort has been made to select problems which do not require substantial background in a specific application area. Instead, understanding the problems we have selected calls for minimal effort. In that way, the reader can concentrate on understanding the solution approach and the solution program. For anyone who uses a scientific calculator, the problems in the applications portion of this book should be both useful and easily understood.

It should be noted that all sample problems have been performed on a CORVUS 500. The APF and OMRON calculators should perform identically. Other RPN calculators will not behave in precisely the same way. However, sufficient similarity does exist between all RPN calculators to render most of the solution programs useful. Appendix B describes a technique to adapt the solution programs to other RPN calculators.

ABOUT THE CORVUS 500

The CORVUS provides a large number of calculating functions with a minimum number of keys. Some of the features of this powerful calculating instrument are listed below.

Display Control – Calculated results may be displayed in either of two modes: business mode and scientific notation mode. Floating point and fixed point is available in both modes.

Accuracy – Regardless of the number of digits displayed, the CORVUS internally maintains 12 significant digits. For certain calculations, such as powers, roots and trigonometric functions, the 2 or 3 least significant digits may be incorrect. Even if such "inaccuracies" are encountered, the precision of the CORVUS will almost certainly exceed the precision of the data entered.

Range – The CORVUS will accept entry of values between $\pm 9.99999999999 \times 10^{99}$ and $\pm 0.1 \times 10^{-99}$. However, some functions are only defined for certain values and other functions utilize approximations that are relatively inaccurate in certain ranges. These restrictions are summarized in Appendix A.

The CORVUS offers an expanded range for displaying results. Calculations generating results that are not within the normal operating range of the CORVUS can be obtained if they fall between $\pm 9.99999999999 \times 10^{99}$ and $\pm 9.99999999999 \times 10^{199}$ or between $\pm 0.1 \times 10^{-99}$ and $\pm 0.1 \times 10^{-199}$. The effective range for calculated results on the CORVUS is therefore $\pm 10^{-200}$ to $\pm 9.99999999999 \times 10^{199}$. Although calculated results are valid for this entire range, a result out of normal range (0.1×10^{-99} to $9.99999999999 \times 10^{99}$) will not be valid for use in further calculations.

Error Indication – The display will flash to indicate out of range results or undefined function arguments.

Memories – The CORVUS has 10 addressable memories plus a special purpose memory. Additionally, the CORVUS has a 4 level memory stack for temporarily storing operands during calculation sequences.

Display Key – This key, marked [DSP], serves two purposes. First, it is used in all display control sequences. Second, it serves as a second function or shift key. Over many of the keys a second function appears in gold letters. The display key is utilized to perform these functions.

Inverse Key – The utility of each key is further increased through the use of the inverse key. The inverse of nearly every function is available.

PART I: HOW TO USE YOUR CALCULATOR

1 USING PART I

This first part of the book is intended to introduce you to your calculator. Each calculator function is described and simple examples are presented.

The examples employed in Part 1 are straightforward function applications. They are constructed in simple steps with each keystroke outlined, as well as indicating the accompanying display.

Keystrokes for these examples are shown as [FUNCTION]. For example, the key labeled y^x is shown as [Y^x]. Shifted keystrokes are shown in the same manner. Thus $\sqrt{x}$ becomes [$\sqrt{x}$] and the function $\sqrt{x}$ is indicated by [DSP] [$\sqrt{x}$].

All of the problems in Part 1, unless otherwise noted, will assume that the calculator has just been turned on. That is, all entries are displayed to the nearest hundredth and all memories are zeroed.

Use of the calculator is carefully developed in this part of the book. Our goal is to enable you to build upon these basic operating instructions and, with the aid of the application problems, to enable you to extend the calculator's powerful features to suit your own calculating requirements.

Even those who are already familiar with the basic operation of the calculator should not ignore this part of the book. We think there may be a few features that will come as a pleasant surprise.

2 ENTERING AND DISPLAYING DATA

2.1 Display Formats

The CORVUS 500 has two basic display formats – business and scientific. Business format displays 12 digits and sign (-). Scientific notation displays a 10 digit number part, or mantissa, which is multiplied by a power of ten, or exponent part. Both the mantissa and the exponent can be negative (-).

2.2 Entering Data

Data can also be entered in either a business or scientific format.

2.2.1 Entering Data in Business Mode

In business mode, positive numbers are merely keyed-in. Twelve digits can be entered and any digit key pressed after the twelfth digit is ignored. To enter negative numbers, the CHANGE-SIGN key (CHS) is utilized. CHS is effective at anytime after the first digit of the number has been keyed. The DECIMAL POINT key (.) can be depressed at any spot desired in order to enter a decimal point.

2.2.2 Entering Data in Scientific Notation

In scientific mode, the mantissa part is entered in the same manner as a number in business mode. The exponent part is inserted by depressing the ENTER EXPONENT key (EE). Up to a 10 digit mantissa can be displayed in scientific notation mode. If an 11 or 12 digit mantissa has been entered, the last digits are internally maintained, but not displayed. To enter a negative exponent, CHS is pressed at any point after EE has been depressed.

Only a 2 digit exponent can be entered. If more than 2 digits are keyed-in, the last 2 digits are retained (i.e., key-in: 3.69 EE 963; the mantissa is 3.69, the exponent is 63). If no mantissa has been entered, then when EE is pressed the calculator automatically assumes a mantissa value of one.

2.3 Clearing Data

The CLEAR X button (CLX) automatically zeroes the display. However, if you have keyed-in an improper exponent in scientific notation, and do not wish to clear the entire entry, merely continue to key-in the digits desired. Only the last two digits of the exponent inserted are maintained (see 2.2.2). Clearing the calculator's stack and memories is discussed in sections 4 and 5, respectively.

2.4 Display Control

The CORVUS 500 can be formatted to display entered data and calculated results in two modes, business and scientific. Display controls do not impact data as keyed. In the examples to be given, one value (1234.56789123) is entered and merely reformatted in each example.

2.4.1 Basic Display Control

There are two basic display control sequences – DSP SCI for scientific notation and DSP INV SCI for business mode. In scientific mode, up to a 10 digit mantissa can be displayed, but extraneous zeroes are suppressed. In

business mode 12 digits can be displayed, but once again, extraneous zeroes are suppressed.

Examples: (reformatting 1234.56789123)

KEY	DISPLAY	COMMENT
DSP SCI	1.234567891 03	Scientific mode
DSP INV SCI	1234.56789123	Business mode

2.4.2 Rounding Options

In both modes, the mantissa can be rounded to a fixed number of decimal places by keying-in DSP and then the number of decimal places to be displayed. In business mode with round-off, no number can be displayed to more than nine decimal places. For large numbers the calculator can display as many decimal places as will fit on the 12 digit display. Thus the number of decimal places displayed may be limited to the number of digits remaining after displaying the digits of the integer (non-fraction) part of the number. In scientific mode, the calculator can always display up to nine decimal places.

The calculator rounds up one if the first digit not being displayed is greater than or equal to 5. It is important to note that although the display is rounded to the desired number of decimal places, the calculator still internally maintains all 12 digits.

Examples: (reformatting 1234.56789123)

KEY	DISPLAY	COMMENT
DSP INV SCI	1234.56789123	Business mode
DSP 3	1234.568	Rounds up one in last digit
DSP 9	1234.56789123	Only able to display 8 decimal place
DSP SCI	1.23456789 03	Scientific Notation
DSP 3	1.235 03	Rounds up one in last digit
DSP 9	1.234567891 03	Rounds to 9 decimal places

2.4.2.1 Predefined Format

When a CORVUS 500 is switched "on", it automatically displays in business mode with 2 decimal places shown. This is equivalent to the display format caused by keying [DSP] [INV] [SCI] [DSP] 2.

2.4.3 Automatic Conversions

Regardless of the display mode, numbers may be entered in either format. When in business display mode, a number may be entered in scientific form and it will be automatically converted to business form during the calculation sequence. If in scientific display mode, a number entered in business mode will be automatically converted to scientific form. In the course of a problem, the form of the operands may be intermixed; each operand may be entered in the most convenient form, with any necessary conversion being performed automatically.

When the calculator is set in business mode, certain automatic display format conversions occur. When calculated answers or data entered fall outside of a predefined range the display converts to scientific mode. Only numbers between ±.000000000001 and ±999999999999 can be displayed in business mode. Any data out of that range (which is still within the calculator's range. See section 2.6) will be displayed in scientific notation. It will continue to be rounded to the same number of decimal places as previously formatted. When the magnitude of the value to be displayed is too small to be displayed in the formatted number of decimal places, the data will be displayed in scientific notation. As long as the display hasn't been reformatted to scientific mode, the calculator will convert back to business mode should the data once again fall within the given range. It should also be noted that while the calculator is formatted to scientific notation, no conversions to business mode are required and none will occur. The example given below illustrates this automatic conversion feature. The ENTER key ([ENT]) can be viewed as part of the mechanism for effecting formatting of data.

Example:

KEY	DISPLAY	COMMENT
[DSP] [INV] [SCI]	0.	Business mode
[DSP] 2	0.00	2 decimal place format
.003 [ENT]	3.00 -03	Automatic conversion because value too small for display format
.03 [ENT]	0.03	Back to business mode with round off to two decimal places

2.5 Error Indication

Under certain circumstances, such as dividing by zero or a calculated result being out of the calculator's normal range, the calculator indicates an error. The error is shown by a flashing display. Depressing [CLX] stops the flashing and a second depression clears the display. While the display is flashing, the calculator is, in effect, locked. In order to "unlock" the calculator [CLX] must be pushed. Appendix A summarizes error conditions.

2.6 Extended Calculated Range

The normal range of the CORVUS 500 is from $\pm.1 \times 10^{-99}$ to $\pm 9.99999999999 \times 10^{99}$. However, calculated results can be obtained between $\pm 1 \times 10^{100}$ and $\pm 9.99999999999 \times 10^{199}$ and between ±.

1×10^{-199} and $\pm.1 \times 10^{-99}$ and $\pm.1 \times 10^{-199}$.

Therefore, the effective range of the calculator is between $\pm.1 \times 10^{-199}$ and $\pm 9.99999999999 \times 10^{199}$.

If the result is outside the normal range yet within the extended range, the display will flash. Upon depression of [CLX], however, the flashing ceases and a valid result is displayed. In these extended range results the most significant digit of the exponent part is not displayed. That is, the exponent is really between either 100 and 199 or -100 and -199 although only the last two digits and sign (-) are shown (e.g., when a flashing 6.721354687 27 is displayed, the result is actually $6.721354687 \times 10^{127}$. The mantissa part of results which are out of normal range is valid for use in further calculations. The exponent part of these results, however, is off by 100 (i.e., the value displayed is the value now internally maintained by the calculator.). If $\pm 10.00 \times 10^{99}$ (± 9.99999999999 with no round-off) or $\pm 0.10^{-99}$ is displayed flashing, the result is out of the extended range, too large or too small, respectively.

3.1 Why RPN?

Three forms of representation for arithmetic expressions have come into common use. We are all familiar with one of these notations – ordinary notation, which is called algebraic by calculator manufacturers and called infix by mathematicians and computer scientists. The term infix refers to the position of the operator in relation to the operands. For example, in the expression a + b, the operator + is fixed between the operands a and b.

Its familiarity would seemingly make infix notation an ideal choice for all calculators. Unfortunately, infix has certain inadequacies. Consider the expression 2 × 3 + 4 × 5. Is this expression equivalent to 2 × 7 × 5 = 70 or to 6 + 20 = 26?

The ambiguity could be resolved with parentheses. Thus the original expression might be written as 2 × (3 + 4) × 5 or as (2 × 3) + (4 × 5). A little experience with a parentheses-dependent notation should be sufficient to indicate how error prone such an approach is. Both missing and extraneous parentheses are sure to be annoyingly frequent.

Alternatively, the ambiguity could be resolved by defining an operator hierarchy. For example, we could define a hierarchy in which multiplications are performed before additions. Thus the original expression would be interpreted such that 2 × 3 + 4 × 5 = 6 + 20 = 26. For expressions involving sum-of-products, this hierarchy works very well. If the expression to be evaluated is not a sum-of-products however, we are forced to use parentheses or to reformat the expression. Therefore to override the sum of products hierarchy, we could do 2 × (3 + 4) × 5 or 2 × 3 × 5 + 2 × 4 × 5 or 3 + 4 = 7 and 7 × 2 × 5.

The inadequacies of infix (or algebraic) notation become more severe as the complexity of the expression increases and are sufficient to cause consideration of two other expression notations. These notations are known as "Polish" because they were first described by the Polish mathematician Lukasiewisz. Fortunately (and understandably) these notations are not known as Lukasiewisz notations. The two varieties of Polish notation are called postfix (or reverse) and prefix Polish. Both permit parentheses-free, unambiguous representation of arithmetic expressions. In postfix notation the operator follows its operands, thus the infix expression 2 + 3 is written as 2 3 +. In prefix notation, the operator precedes its operand thus we have + 2 3. The table below shows the expression 2 × 3 + 4 × 5 in infix, postfix and prefix notations.

INFIX	POSTFIX	PREFIX
(2 × 3) + (4 × 5)	2 3 × 4 5 × +	+ × 2 3 × 4 5
2 × (3 + 4) × 5	2 3 4 + × 5 ×	× × 2 + 3 4 5

Both prefix and postfix notation eliminate the ambiguity of infix without parentheses. Typically however, expressions are written in infix notation. Fortunately the conversion from infix to postfix (RPN) is very simple.

1. Start at the left of the expression.
2. Record next operand.
3. If an operation can be performed (i.e., if all necessary operands have been recorded), record the operator.
4. If another operation remains which can be performed, repeat steps 3 and 4.
5. If more operands and operators remain in the expression, then repeat steps 2, 3, 4, and 5.

The conversion from infix notation to prefix notation is not nearly so simple:

1. Locate the operation to be performed last, record the operator.
2. Locate the leftmost operand of the operator in 1.
3. If the operand is itself an expression, follow all steps in this procedure for that sub-expression.
4. If the operand is instead a data item, record it.
5. If there exists a next leftmost operand, repeat steps 3, 4 and 5.

RPN offers the advantage of unambiguous representation of complex expressions. Furthermore, as one would suspect from the simple conversion between infix and RPN, RPN is easily mastered. RPN is therefore best suited to meet your calculation requirements. The next two subsections should help you to feel comfortable with RPN and to lay the groundwork for effective utilization of your calculator.

3.2 RPN and Your Calculator

To perform calculations on your RPN calculator, operands must be keyed-in before the operator. Thus the calculator requires some internal means for storing operands pending operator entry. In the calculator, this is accomplished by a set of registers called the stack. A register is an electronic element utilized for temporary data storage. Since multiple operands may be keyed in before the operator, some means to indicate the end of an operand is needed (infix uses the operator itself to separate operands). The ENTER key

provides this means. The stack and operations on the stack will be discussed in section 4. For purposes of this section we will disregard the potential for exceeding the stack's capacity and will delay discussion of the "internal" workings of the stack. In essence we can conceive of the stack as a place to temporarily store operands – a storage facility with the special property that the last operand stored is the first operand returned.

3.3 RPN and Four Basic Arithmetic Operations

This subsection presents a series of examples illustrating calculations involving addition, subtraction, multiplication and division. After keying in each number, either the ENTER key ([ENT]) or the appropriate operator key (+ , - , × , ÷) is depressed. For now, the ENTER key can be thought of simply as a mechanism for indicating the end of a number entry which is to be stored in the stack.

Example: Turn your calculator on and let's try one variety of our familiar example. – (2 × 3) + (4 × 5) ?

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Data	*2*	2	Key in first operand
2	Enter	[ENT]	2.00	Store operand (2) on stack
3	Data	*3*	3	Key in next operand
4	Multiply	[X]	6.00	2 X 3 is displayed
5	Data	*4*	4	Key in next operand
6	Enter	[ENT]	4.00	(2 X 3 and 4 are now on stack
7	Data	*5*	5	Key in last operand
8	Multiply	[X]	20.00	4 X 5 is displayed
9	Add	[+]	26.00	(2 X 3) and (4 X 5) were on stack and are now added, displaying results.

* Notice that we have not used any memories yet.

Example: The other variety of our familiar example 2 × (3 + 4) × 5 ?

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Data	*2*	2	Key-in first operand
2	Enter	[ENT]	2.00	Store operand on stack
3	Data	*3*	3	Key-in next operand
4	Enter	[ENT]	3.00	Store operand on stack
5	Data	*4*	4	Key-in next operand
6	Add	[+]	7.00	3 + 4 is displayed
7	Multiply	[X]	14.00	2 X (3 + 4) is displayed
8	Data	*5*	5	Key-in next operand
9	Multiply	[X]	70.00	2 X (3 + 4) X 5 is displayed

Example: $\frac{[(16 \div 4) - 1.5] + 15 - 3}{17 - (1.5 \times 9.8)}$?

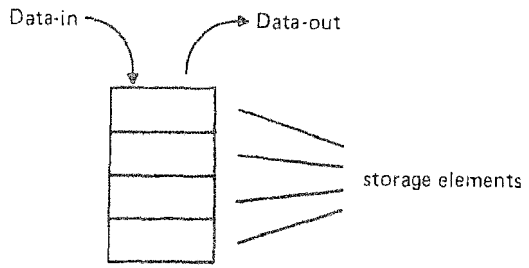
STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Data	*16*	16	Key-in first operand
2	Enter	[ENT]	16.00	Store operand on stack
3	Data	*4*	4	Key-in next operand
4	Divide	[÷]	4.00	16 ÷ 4 is displayed
5	Data	*1.5*	1.5	Key-in next operand
6	Subtract	[–]	2.50	[(16 ÷ 4) - 1.5] displayed
7	Data	*15*	15	Key-in next operand
8	Add	[+]	17.50	[(16÷4)-1.5]+15 displayed
9	Data	*3*	3	Key-in next operand
10	Subtract	[–]	14.50	[16÷4)-1.5]+15-3 displayed
11	Data	*17*	17	Key-in next operand
12	Enter	[ENT]	17.00	Store operand on stack
13	Data	*1.5*	1.5	Key-in next operand
14	Enter	[ENT]	1.50	Store operand on stack
15	Data	*9.8*	9.8	Key-in next operand
16	Multiply	[X]	14.70	1.5 X 9.8 displayed
17	Subtract	[–]	2.30	17 - (1.5 X 9.8) displayed
18	Divide	[÷]	6.30	[(16÷4)-1.5]+15-3 and 17-(1.5 X 9.8) were on stack. Result of division now displayed

4 THE STACK

4.1 The Stack Concept

The stack is a storage structure whose application is not limited to RPN calculators. To be more precise, the stack is a well-developed conceptual structure which is frequently encountered in data processing and which appears in a somewhat modified form in calculators. We feel that the peculiarities of the calculator's stack are more easily understood if viewed in the context of a "true" stack.

A stack is a set of storage registers forming a linear list (see Figure 4-1). Insertions and deletions are made at only one end of this list. Thus, the stack is called a *last-in-first-out* or LIFO list because the last item added to the list is always the first item removed from the list.



Special terminology is utilized in referring to the stack. Data is inserted onto the *top* of the stack and removed from the top of the stack. The *bottom* of the stack is not immediately accessible – all other items must first be removed. When an item is added to the stack, we *push* the stack. The top remains in the same location but its contents are changed. The old top is moved down one location. When an item is removed from the stack, we *pop* the stack.

Stack terminology comes from an analogy with the spring-loaded stack of plates frequently found in cafeterias. When an additional plate is added to the stack, the weight of the new plate pushes the rest of the stack downward. The new plate is at the same level as the previous top plate. When a plate is removed from the stack, the rest of the stack pops upward. The top of the stack always remains at the same level.

4.2 Calculator Stack Operations

The CORVUS 500 stack consists of four storage registers. These registers are labeled X, Y, Z, and W, where X is the top of the stack and W is the bottom of the stack. Register W (the bottom of the stack) is sometimes called register T. After each operation is completed and as new data is being keyed-in, the value stored in X and the value displayed are the same.

Each function performed by the CORVUS requires either one or two operands. When a function is entered, these operands are found on the top of the stack. Each function performed by the CORVUS produces one or two results. The net effect of each function is to pop operands off the stack and push results onto the stack. The description of each function will include the effect of the function on the stack.

The subsections which follow describe the stack operations provided by the CORVUS. Each operation is described and the keystrokes to perform the operation are specified. Each subsection also presents an example of the use of the operation in tabular form. Rows in these tables represent the content of the register indicated in the leftmost column. Columns in these tables represent the content of the registers after the keystroke sequence heading the column has been completed. The values in the tables are all integers. (DSP INV SCI DSP 0 will cause the display to show only integers.) The tables are sequential; each proceeding from the state of the previous table.

4.2.1 Modified Push (or Enter)

This operation causes the X register to be copied and pushed into the Y register. The value in the X register remains the same. The previous value in the Y register is placed in the Z register and the previous value in the Z register is placed into the W register. If a value was in the W register before the modified push operation, the value is lost. The modified push operation is performed by depressing the ENTER key (ENT).

Example Sequence:

R \ K	1	ENT	2	ENT	3	ENT	4	ENT	5
X	1.	1.	2.	2.	3.	3.	4.	4.	5.
Y	0.	1.	1.	2.	2.	3.	3.	4.	4.
Z	0.	0.	0.	1.	1.	2.	2.	3.	3.
W	0.	0.	0.	0.	0.	1.	1.	2.	2.

It is important to note that after the ENTER key is depressed, the data item next keyed in will write over the old X register value.

4.2.2 Push

Two variations of the "true" stack push operation occur on the CORVUS. First, the combination of a modified push and a data value being keyed in is equivalent to a push. This form of push is obviously the purpose of the ENTER key. As was discussed in Section 3, some means is necessary to distinguish between the continuation of a data entry and the beginning of a new data entry.

The second variation of the push operation does not require a keystroke. Virtually every operation leaves a push pending. When a number is entered following an operation, it is pushed onto the stack. The ENTER key is not required. The only function key which does not cause a push on the next number entry is the summation key [Σ+] , whose use is discussed in Section 20.

Example Sequence:

K / R	Previous Values	* [+]	*** 10	[ENT] 2	* [÷]	* [−]	*** 6	[ENT] 5	* [×]
X	5.	9.	10.	2.	5.	4.	6.	5.	30.
Y	4.	3.	9.	10.	9.	3.	4.	6.	4.
Z	3.	2.	3.	9.	3.	3.	3.	4.	3.
W	2.	2.**	2.	3.	3.**	3**	3.	3.	3.**

*The four basic arithmetic operations all have an identical effect upon the stack. The two operands are popped off the stack and the result is pushed onto the stack.

**The CORVUS stack has a special property. Whenever the stack is popped, a copy of the W register value remains in the W register.

***The value is pushed onto the stack since the operation was performed immediately before the number entry.

4.2.3. Modified Pop (or Clear X) and Pop

The modified pop operation causes the X register to be zeroed and causes the next value entered to be placed in the X register without affecting the rest of the stack. Naturally, this operation is most useful for eliminating erroneous entries. A modified pop is caused by depressing the CLEAR X key [CLX] .

A true pop operation can be caused by either [CLX] [+] or [CLX] [−]. In both cases the X register is first zeroed and the value in the Y register is not changed (but is moved to the X register) by the operation that follows. The net result of the two keystrokes is to pop the stack.

Example Sequence:

K / R	Previous Values	[CLX]	2	[ENT] 5	[CLX]	* [+]	[CLX]	* [−]	** 5
X	30.	0.	2.	5.	0.	2.	0.	4.	5.
Y	4.	4.	4.	2.	2.	4.	4.	3.	4.
Z	3.	3.	3.	4.	4.	3.	3.	3.	3.
W	3.	3.	3.	3.	3.	3.	3.	3.	3.

*The sequences [CLX] [+] and [CLX] [−] pop the stack.

**The sequence shown ([CLX] [−] followed by data entry) and the equivalent sequence with [CLX] [+] followed by data entry have the same net effect upon the stack as simply [CLX] followed by data entry.

The clear key, [CLR] , is used to zero the entire stack. The appropriate keystroke sequence is [DSP] [CLR] .

4.2.4. Rolling the Stack

At times, one may need to review the contents of the entire stack or to shift data within the stack. The ROLL key ([R↓]) provides these capabilities. The roll operation causes the top of the stack to be inserted on the bottom of the stack and causes the stack to be popped. That is, after depressing [R↓]) the contents of the X register are moved to the W register, the contents of the Y register are moved to the X register, the contents of the Z register are moved to the Y register, and the contents of the W register are moved to the Z register. In this operation the stack can be viewed as a circularly connected list in which the top "rolls" around to the bottom slot and each of the other positions "roll" up one slot. Since the stack has four registers, depressing the [R↓] four times will return the stack to its original position. The roll operation leaves a push pending for the next number entry.

The following table illustrates the roll operation. The stack is first cleared and then four new values are pushed onto the stack. Four roll operations are then performed.

Example Sequence:

K / R	DSP CLR	1	ENT 2	ENT 3	ENT 4	R↓	R↓	R↓	R↓
X	0.	1.	2.	3.	4.	3.	2.	1.	4.
Y	0.	0.	1.	2.	3.	2.	1.	4.	3.
Z	0.	0.	0.	1.	2.	1.	4.	3.	2.
W	0.	0.	0.	0.	1.	4.	3.	2.	1.

4.2.5 Exchange Registers

When coupled with the ROLL key, the EXCHANGE key (X⇔Y) provides the capability of completely reorganizing data stored in the stack. The exchange operation moves the contents of the X register into the Y register and the contents of the Y register into the X register. This operation is particularly useful when operands have inadvertently been entered out of order. For example, a divisor may be in the Y rather than the X register.

The following table illustrates the exchange operation. The registers are initialized with the values from sub-section 4.2.4. The keystroke sequence utilizes the roll and exchange operations to reverse the order of the registers' contents.

Example Sequence:

K / R	Previous Values	X⇔Y	R↓	R↓	X⇔Y
X	4.	3.	4.	2.	1.
Y	3.	4.	2.	1.	2.
Z	2.	2.	1.	3.	3.
W	1.	1.	3.	4.	4.

5 MEMORY

The CORVUS 500 has several electronic storage registers called memories.

The stack is actually four memories. There are also a number of memories which are completely invisible to the user of the calculator. For example, the X register and the display register are physically distinct although they appear logically as a single register. Other "invisible" memories are utilized as a scratchpad during calculations. These "invisible" memories are termed not addressable since there is no way for the calculator user to access them. In addition to the four stack memories, the calculator has 11 other addressable memories. Ten of these are called named memory, and one is a special, unnamed memory used during calculation sequences involving the summation function (see Part I, Section 20). The named memories are memory 0, 1, 2, 3, 4, 5, 6, 7, 8, or 9. Memory 0 is also called LAST X. The appropriate digit key is used in accessing these memories. The following five subsections describe the operations available for the ten named memories. Section 20 of Part I describes operation of the unnamed memory.

5.1 Storing and Recalling Data

The STORE and RECALL keys are used in manipulating the named memories. The INVERSE key doubles as the STORE key. This multiplicity of purpose is indicated by the gold letters STO. Operations involving the STORE key do not involve the INVERSE key. Therefore the DISPLAY key (DSP) is not needed to distinguish between the two uses of the INVERSE/STORE key. The "little gold box" enclosing the gold letters indicates that no shifting via a DSP is needed. Similarly, the RECALL key (RCL) requires no shifting.

Values are stored by depressing STO followed by the memory name (also called address). Values are recalled from memory by depressing RCL followed by the memory name. This example should clarify the storing and recalling operations – switch on your calculator and try a few simple memory operations.

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Data	*3.55*	3.55	Put in a data entry.
2	Store	STO 7	3.55	Store 3.55 in memory 7—leave a push pending.
3	Data	*6.15*	6.15	Put in a second data entry.
4	Store	STO 4	6.15	Store 6.15 in memory 4.
5	Multiply	X	21.83	Calculate 3.55 x 6.15.
6	Recall	RCL 7	3.55	Recall operand from memory—leave a push pending.
7	Recall	RCL 4	6.15	Recall other operand.
8	Multiply	X	21.83	Repeat multiply.
9	Store	STO 4	21.83	Store product over old operand.
10	Clear	CLX	0.00	Clear display.
11	Recall	RCL 4	21.83	To no one's surprise, the product we stored is still there!

5.2 Exchange with Memory

The CORVUS provides an additional memory operation. Depressing either STO RCL or RCL STO followed by a memory name causes the contents of the X register to be exchanged with the indicated memory. The following example illustrates this application.

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Data	*3.14*	3.14	Enter a number.
2	Store	STO 3	3.14	Store the number.
3	Clear stack	DSP CLR	0.00	Clear all values in the stack.
4	Data	*7.93*	7.93	Enter another number.
5	Exchange mem.	STO RCL	3.14	Exchange display and memory 3.
6	Clear stack	DSP CLR	0.00	Clear stack.
7	Recall	RCL 3	7.93	Yes, it was really exchanged!
8	Clear stack	DSP CLR	0.00	Clear stack again.
9	Data	*4.56*	4.56	Enter another number.
10	Exchange mem.	RCL STO 3	7.93	Try another combination for exchange command.
11	Recall	RCL 3	4.56	Yes, it was really exchanged!

5.3 Limitations on Memory

Although the store, recall and exchange memory operations are valid for each of the ten memories, there are certain restrictions to the full use of memories 7, 8, 9, and 0. Memories 7, 8, and 9 are utilized during calculation sequences involving the summation function. If the summation function is not involved in the calculation, then memories 7, 8 and 9 can be treated identically to memories 1 through 6. Section 20 of Part I will describe restrictions to the use of memories 7, 8 and 9 during summation calculations.

Memory 0 is a special memory also called LAST X. This memory is used in nearly every calculation sequence. The LAST X memory is further described in Section 5.5.

5.4 Clearing Memory

A clear memory instruction is neither provided on the CORVUS 500 nor is such an instruction necessary. When a new value is stored in memory, any trace of the value previously stored in that memory is lost. Thus memories are effectively cleared before being stored into. Naturally, you can always clear all memories by switching the calculator off and then on again.

5.5 LAST X

Memory 0 (or LAST X) is a special purpose memory. Although the store, recall and exchange operations are valid for this memory, the value stored in memory 0 is changed throughout calculation sequences. Each time an operation is performed, the last operand to that operation is stored in memory 0. Thus the term LAST X.

The LAST X memory is a useful feature for correcting errors. An incorrect operation can be undone with the aid of LAST X. For example, if you inadvertently multiply by 5 you could correct this error by depressing RCL LAST X ÷ .

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Data	*4.35*	4.35	Enter an operand.
2	Push	ENT *5*	5.	Enter second operand.
3	Multiply	X	21.75	Multiply
4	Stop		21.75	Realize that a mistake has been made—should have divided, not multiplied.
5	Recover	RCL LAST X	5.00	Recover the last operand.
6	Divide	÷	4.35	Undo the erroneous operation.
7	Recall	RCL LAST X	5.00	Retrieve operand again.
8	Divide	÷	0.87	Now we've got the desired result!

6 CHANGE SIGN

The CHANGE SIGN key has already been introduced in Section 2. The purpose of the key as described in that section was to enter negative numbers and negative exponents. The CHANGE SIGN key also provides an unary-minus function. Unary-minus permits the change of sign for calculated results.

The operation is referred to as unary because it requires only one operand. The four basic arithmetic operations already discussed (+, -, ×, ÷) are termed binary functions because they require two operands. All operations available on the CORVUS are either unary or binary.

Example: $-\left(\frac{9.67 \times (-1.45)}{3.42 \times 10^{-6}}\right) \stackrel{?}{=}$

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Format Display	DSP SCI DSP	0.00 00	Set display for scientific notation.
2	Data	*9.67*	9.67	
3	PUSH	ENT *1.45* CHS	-1.45	Enter negative operand.
4	Multiply	×	-1.40 01	Obtain 9.67 x (-1.45)
5	Data	*3.42* EE *6* CHS	3.42 -06	Enter operand with negative exponent.
6	Divide	÷	-4.10 06	Obtain $\frac{9.67 \times (-1.45)}{3.42 \times 10\text{-}6}$
7	Unary-Minus	CHS	4.10 06	

7 PI

Constants are extremely useful in many calculation sequences. Appendix C contains a number of frequently used constants.

Pi may very well be the most commonly used constant. For convenience, the CORVUS supplies a PI key for entering this constant. The value used is 3.14159265359. By depressing DSP π this value is pushed onto the stack. The following simple example calculates the circumference of a circle to illustrate the use of the PI key.

Example: Find the circumference of a circle of radius 3.5 feet. The formula circumference = $2\pi r$ is used.

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Data	*2*	2	Enter constant 2.
2	Pi	DSP π	3.14	Enter constant π
3	Multiply	×	6.28	Find 2π
4	Data	*3.5*	3.5	Enter r
5	Multiply	×	21.99	Obtain $2\pi r$.

8 METRIC⇌ENGLISH CONVERSIONS

The metric system of units is a concise and consistent way of expressing amounts of length, volume, mass (weight), and temperature. The system employed most often in the United States is called "English" although it is so outmoded even the English no longer use it. The metric system, now in use throughout most of the world, employs a system of basic units (e.g., liters is the basic unit of volume) and a series of prefixes which represent powers of ten (listed in Section 8.1.2). The United States is currently in the process of switching to the metric system and for that reason, many measurements are only given in one system or the other. Thus a simple method of conversion between metric and English systems is required. The CORVUS 500 provides this method. Length, volume, mass, and temperature conversions can be easily accomplished with the calculator.

8.1 The Metric System

8.1.1 Basic Units of the Metric System

In any system of measurement it is necessary to begin with some quantities that are considered to be elemental, or basic. The choice of these particular quantities is arbitrary.

The four elemental units of the metric system* which we are concerned with are:

UNIT	ABBREVIATION	UNITS OF	CONVERSION TO ENGLISH
Meter	m	length	1m = 39.37000787402 inches
Gram	g	mass	1g = .002204622622 pounds
Degree Centigrade	°C	temperature	temperature C= temperature (F -32) x 5/9.
Liter	ℓ	volume	1ℓ=.264179449175 gallon

*Not to be confused with the SI system of units, which includes units for time, amount of a substance, electrical current, and luminence of light.

8.1.2 Prefixes Utilized in the Metric System

In the metric system, all measurements which need to be expressed in units larger or smaller than the basic unit are expressed in a unit that is formed from the basic unit. A prefix which represents a power of ten (positive or negative) is affixed to the basic unit. (e.g., kilo represents 10^3; a kilogram is equivalent to 1000 grams). Through this method, any amount, no matter the size, can be easily expressed. The same prefixes are used for all of the basic units. Following is a list of the metric prefixes, and their corresponding value and abbreviation.

PREFIX	ABBREVIATION	VALUE
tera-	T	10^{12}
giga-	G	10^{9}
mega-	M	10^{6}
kilo-	k	10^{3}
hecto-	h	10^{2}
deca-	de	10^{1}
deci-	d	10^{-1}
centi-	c	10^{-2}
milli-	m	10^{-3}
micro-	μ	10^{-6}
nano-	n	10^{-9}
pico-	p	10^{-12}

8.2 Conversions on the Calculator

The CORVUS 500 can perform conversions of temperature, volume, length, and weight between the English and metric systems. Specifically, the calculator is programmed for conversions between degrees Centigrade and degrees Fahrenheit, between liters and gallons, between centimeters and inches and between kilograms and pounds. Because of the simplicity of the metric system, any metric unit can be easily converted to the units provided by the calculator (e.g., convert milliliters to liters merely by multiplying by 10^{-3}). However, conversions between English units are more complicated, and a table of units is required. A table of English equivalents can be found in Appendix C.

8.2.1 Metric→English Conversions

The four metric→English conversion keys are located on the shift of the four basic arithmetic function keys (÷ , × , - , +). The keystroke sequence for these conversions is [DSP] [CONVERSION FUNCTION] . This sequence converts the value in the X-register and leaves the new value in the X-register with a push pending.

8.2.1.1 Metric→English Conversions of Temperature

The calculator automatically converts degrees Centigrade, or Celsius, to degrees Fahrenheit. The values are computed utilizing the formula °F = (9/5 °C) + 32. The value to be converted is keyed into the X-register and followed by the keystroke sequence [DSP] [C→F] . It should be noted that a third scale of temperature is often employed in the scientific world, the Kelvin, or absolute, temperature scale. The conversion from degrees Kelvin to degrees Centigrade is accomplished by subtracting 273.16.

Example: Convert 267.83 °K. to °F.

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Data	*267.83*	267.83	Enter data °K
2	Subtract	[ENT] 273.16 [−]	-5.33	Obtain °C
3	Degree C to Degree F	[DSP] [C→F]	22.41	Compute value °F

8.2.1.2 Metric→English Conversions of Volume

The calculator automatically converts liters to gallons using the constant 1ℓ = .264179449175 gallon. The constant utilized by any of the conversion functions (excluding temperature, which uses a formula) can be found by keying 1 and then the conversion function key sequence.

Another excellent feature of the metric system is that volume units are derived from the length measurements. Using the formula which states that one cubic centimeter (cm^3 or cc) is equal to one milliliter (ml), the dimensions of a container can be easily converted to a volume unit.

The keystroke sequence for the conversion from liters to gallons is [DSP] [LTR→GAL] .

Example: Convert 8138.41 cm^3 to gallons

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Data	*8138.41* [ENT]	8138.41	Enter cm^3 (same as milliliters)
2	Multiply	[EE] [CHS] *3* [X]	8.14	Convert to liters
3	Lit.→Gal.	[DSP] [LTR→GAL]	2.15	Compute value in gallons

8.2.1.3 Metric→English Conversion of Length

The calculator automatically converts centimeters to inches using the constant 1 cm = .393700787402 inches. The keystroke sequence for conversions from centimeters to inches is [DSP] [CM→IN] .

Example: How long is a 100 meter dash (in yards)?

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Data	*100* [ENT]	100.00	Enter data in meters
2	Divide	[EE] [CHS] 2 [÷]	10000.00	Convert to cm.
3	CM→IN	[DSP] [CM→IN]	3937.01	Obtain value in inches
4	Divide	36 [÷]	109.36	Compute value in yards

8.2.1.4 Metric→English Conversions of Mass

The calculator automatically converts kilograms to pounds using the constant 1 kg = 2.20462262185 lb. The keystroke sequence for conversion from kilograms to pounds is [DSP] [KG→LB].

Example: Upon leaving the United States for Europe, Ron weighed 160 pounds. When he weighed himself in Europe, he found that he weighed 78.30 kilograms. How much weight did Ron gain (in pounds)?

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Enter	*160*	160.00	Original weight in x and y
2	Kilograms to Pounds	*78.30* [DSP] [KG→LB]	172.62	Weight in Europe
3	Exchange	[Y⇔X]	160.00	Weight in Europe is in Y-register Original weight is in X-register
4	Subtract	[−]	12.62	Compute Ron's weight gain

8.2.2 English→Metric Conversions

The four English metric conversions on the calculator utilize the INVERSE key and the conversion keys. The keystroke sequence for these conversions is [DSP] [INV] [CONVERSION FUNCTION] or [INV] [DSP] [CONVERSION FUNCTION]. This sequence converts the value in the X-register and leaves the new value in the X-register with a push pending.

8.2.2.1 English→Metric Conversion of Temperature

The calculator automatically converts degrees Fahrenheit to degrees Centigrade utilizing the formula °C. = 5/9 (°F -32). The keystrokes used for this conversion are [DSP] [INV] [C→F]. To convert degrees Centigrade to degrees Kelvin add 273.16.

Example: Sven is planning a trip to Los Angeles. He checks the L.A. Times and finds that the temperature is running around 80 degrees. After a brief moment of panic (80 degrees Centigrade is equivalent to 176° F.), Sven realizes that the temperature is expressed in Fahrenheit. What is the temperature in Centigrade?

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Data	*80*	80	Data °F
2	°F to °C	[DSP] [INV] [C→F]	26.67	Compute value in °C

8.2.2.2 English→Metric Conversions of Volume

The calculator automatically converts gallons to liters using the constant 1 gal. = 3.78530579544 liters. The key sequence for conversion from liters to gallons is [DSP] [INV] [LTR→GAL]. It should be remembered that 1 ml. = 1 cm^3.

Example: If a container has a 19.25 gallon capacity, what is its volume (in cm^3)?

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Data	*19.25*	19.25	Volume in gallons
2	Gallons to liters	[DSP] [INV] [LTR→GAL]	72.87	Obtain volume in liters
3	Divide	[EE] [CHS] 3 [÷]	72867.14	Obtain volume in cm^3

8.2.2.3 English→Metric Conversions of Length

The CORVUS 500 provides automatic conversions from inches to centimeters using the constant 1 in. = 2.54 cm. The keystroke sequence utilized is [DSP] [INV] [CM→IN].

Example: 50 meter pools are normally used in competition. In an old sports arena, a 50 yard pool is in use; what is the difference between the two pools (in decimeters)?

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Data	*50* [ENT]	50.00	Data in meters
2	Data	*50* [ENT]	50.00	Data in yards
3	Multiply	36 [×]	1800.00	Convert to inches
4	Inches to cm	[DSP] [INV] [CM→IN]	4572.00	Convert to centimeters
5	Multiply	[EE] [CHS] 2 [×]	45.72	Convert to meters
6	Subtract	[−]	4.28	Difference in meters
7	Divide	[÷] [CHS] 1 [÷]	42.80	Difference in decimeters

8.2.2.4 English→Metric Conversions of Mass

The conversion from pounds to kilograms is provided on the CORVUS 500. This conversion utilizes the constant 1 lb. = .45359237 kg. The keystroke sequence for this conversion is [DSP] [INV] [KG→LB].

Example: One store (A) advertises 5 pounds of rice for $2.19. Another store (B) is selling the same type of rice at 2 kilograms for $1.69. Which store has the better buy?

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Data	*2.19*	2.19	Store A data
2	Divide	ENT *5* ÷	0.44	Store A price per pound
3	1.69	*1.69* ENT	1.69	Store B data
4	Data	*2*	2.00	Store B data
5	Kilograms to Pounds	DSP KG→LB	4.41	Number of pounds of rice
6	Divide	÷	0.38	Obtain Store B price per pound
7	Subtract	−	0.05	Compute difference between Store A and Store B

Conclusion: Store B sells rice for 5¢ per pound less than Store A. Therefore, Store B has the better buy.

8.3 A Multi-Step Conversion

Example: An aquarium measures 72 inches in length, is 24 inches wide and is 20 inches high. Assuming that the aquarium is built of materials with density equal to water, what is the capacity (in gallons) and weight (in pounds) of the filled aquarium?

Approach: This problem can be solved by determining the volume of the aquarium and expressing the volume in cubic centimeters. The volume can then be translated to liters and then to gallons. The volume can also be translated to kilograms (recall that 1 cc (or ml) of water weighs 1 gram) and then to pounds.

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Data	*72*	72	Length in inches
2	Data	ENT *24*	24	Width in inches
3	Data	ENT *20*	20	Height in inches
4	Multiply	X X	34560.00	Volume in cubic inches
5	Data	1	1	
6	In→Cm	DSP INV CM→IN	2.54	Conversion factor cm/in
7	Multiply	ENT ENT X X	16.39	Obtain cc/in^3
8	Multiply	X	566336.93	Obtain volume in cc
9	Divide	EE 3 ÷	566.34	Obtain volume in liters
10	Store	STO 1	566.34	Store a copy for Step 12
11	Lit→Gal	DSP LTR→GAL	149.61	Obtain aquarium capacity in gals.
12	Recall	RCL 1	566.34	Recall volume in liters = weight in kilograms.
13	KG→LB	DSP KG→LB	1248.56	Obtain weight in pounds

Aquarium weighs 1248.56 pounds when filled with water and holds 149.61 gallons.

9 RECIPROCALS

The RECIPROCAL key is used to calculate the reciprocal of any value in the X register. A reciprocal, or multiplicative inverse, is the value which is found when one is divided by a particular number, hence the symbol 1/X. The RECIPROCAL key is located on the shift of EE .

The reciprocal of any value in the X register can be found by keying the sequence DSP 1/x . However, the sequence DSP 1/x will cause the display to flash, indicating an error, if the value in the X register is 0. This operation (1/0) results in the display flashing the maximum value (i.e., 10.00 99 with round-off and 9.999999999 99 without round-off).

Example:

$$\frac{1}{\frac{1}{3}+\frac{1}{-8}} \stackrel{?}{=}$$

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Reciprocal	*3* DSP 1/x	0.33	Obtain 1/3
2	Reciprocal	*8* CHS DSP 1/x	-0.13	Obtain 1/-8
3	Add	+	0.21	Obtain 1/3 + 1/-8
4	Reciprocal	DSP 1/x	4.80	Compute $\frac{1}{1/3 + 1/-8}$

10 FACTORIALS

Factorials are used in a variety of application areas. In Part II of this book, for example, factorials appear in the probability section. The factorial of X (written X!) is defined as the product of the integers from 1 to X (i.e., 1×2× . . . ×X). Thus 4! equals 1 × 2 × 3 × 4 = 24. In addition, 0! is defined as equal to 1. Factorials are only defined for non-negative integers.

The factorial of X is obtained on the CORVUS with the keystroke sequence DSP X! . 69 is the largest integer whose factorial lies within the normal range of the CORVUS. 120! is within the extended range.

Example: 10! $\stackrel{?}{=}$

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Data	*10*	10	
2	Factorial	DSP X!	3628800.00	10! is displayed

11 PERCENTAGE CALCULATIONS

Three percentage calculations are available on the CORVUS.

11.1 Simple Percentages

The percentage of a fixed amount can be obtained with the keystroke sequence [DSP] [%] . The amount is placed in the Y register and the percentage desired is placed in the X register. The keystrokes [DSP] [%] leave the Y register unaffected and replace the original value in the X register (call it x_0%) with x_0% of the value in the Y register.

Example : The price per pound of rice was 40¢ but has since been increased by 12.5% What is the new price per pound of rice?

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Data	*.40* [ENT]	0.40	Original price per pound
2	Data	*12.5*	12.5	Enter %
3	Percent	[DSP] [%]	0.05	Find 12.5% of .40
4	Add	[+]	0.45	New price is displayed

11.2 Percentage Difference

The percentage difference between two amounts can be obtained with the keystroke sequence [DSP] [Δ%] . The amount in the Y register is used as a base. The percentage difference between the amount in the Y register and the amount in the X register is calculated by depressing [DSP] [INV] . The inverse button ([INV]) is ignored if pressed. The calculated value ($\frac{x-y}{y}$ 100) replaces the old value in the X register.

Example: A runner accustomed to competing in the mile, must now compete in 1500 meter races. What is the percentage difference between the two distances?

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Data	*5280*	5280	Enter mile in feet
2	Data	[ENT] *1500*	1500	Enter 1500 meters
3	Multiply	[ENT] [EE] 2 [X]	150000.00	Convert 1500 meters to centimeters
4	CM→IN	[DSP] [CM→IN]	59055.12	Convert to inches
5	Divide	12 [÷]	4921.26	1500 meters in feet
6	% difference	[DSP] [Δ%]	-6.79	1500 meters is 6.79% shorter than mile

11.3 Gross Profit Margin

The CORVUS provides a direct means for computing gross profit margin. Gross profit margin is given by:

Let C = cost

P = Gross profit margin (amount) $P = \frac{RC}{100 - R}$

R = Gross profit percentage or markup % based on selling price

Selling price is equal to C + P.

Gross profit margin is obtained with the keystroke sequence [DSP] [INV] [%] or with [INV] [DSP] [%] . The value in the Y register is assumed to be cost, C, and is unchanged by the gross profit margin operation. The value in the X register is interpreted as the markup percentage based on selling price, R, and is replaced by the gross profit amount R. The selling price C + P can be obtained by pressing [+] after the gross profit margin operation is complete.

Example: A coffee merchant receives notice that the wholesale price of Java beans is being increased to $2.10 per pound. If the merchant sells coffee at only a 25% markup based on selling price, what will be the new selling price of the Java?

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Data	*2.10*	2.10	Enter price per lb. of coffee
2	Data	[ENT] *25*	25	Enter % markup
3	GPM	[DSP] [INV] [%]	0.70	Obtain gross profit margin
4	Add	[+]	2.80	Obtain new selling price

12 SQUARE AND SQUARE ROOT

The SQUARE ROOT key can be used to calculate either the square or square root of the value in the X register. The calculator can only find the square root of positive numbers. The SQUARE ROOT key is located on the shift of [Y^x].

Both of the operations performed by the SQUARE ROOT key, square and square root, can also be performed by the POWER key, whose functions are described in the following section. The square of a number can also be found by keying the number and then [ENT] [X]. These alternate operations require 2 stack levels. The square root of any positive value in the X register can be found by keying the sequence [DSP] [$\sqrt{x}$]. The sequence to find the square of any value in the X register is [DSP] [INV] [$\sqrt{x}$]. It is advantageous to follow those sequences, not only because it requires fewer keystrokes than are necessary with the POWER button, but also because only the X register is required.

Example: $\sqrt{17.2^2 - 217.23} \stackrel{?}{=}$

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Square	*17.2* [DSP] [INV] [$\sqrt{x}$]	295.84	Obtain 17.2^2
2	Subtract	*217.23* [−]	78.61	Obtain 17.2^2 - 217.23
3	Square Root	[DSP] [$\sqrt{x}$]	8.87	Compute $\sqrt{17.2^2 - 217.23}$

The area of a circle is computed with the formula: πr^2

Example: If the radius of a circle is 4.925 inches, what is the the area of the circle? (in square inches)

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Format Display	[DSP] [INV] [SCI] *3*	0.000	Set 3 decimal places display.
2	Pi	[DSP] [π] [DSP]	3.142	Constant Pi
3	Square	*4.925* [DSP] [INV] [$\sqrt{x}$]	24.256	Obtain r^2
4	Multiply	[X]	76.201	Compute πr^2

13 POWERS AND ROOTS

The POWER key can be used to raise a positive number to any finite power or to find any root of a positive number. The calculator cannot find powers or roots of negative numbers or zero and the display will flash zero when calculations are attempted with such values.

To find the power of a number, the number should be pushed into the Y register and the power should be keyed into the X register. The POWER key ([Y^x]) is then depressed to complete the operation. The keystrokes for roots are similar. The number is keyed and pushed into the Y register. The root to be taken is then keyed into the X register. The root operation is concluded with the sequence [INV] [Y^x].

Example: $(8.3)^{3.4} - \sqrt[2.7]{243.61} \stackrel{?}{=}$

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Data	*8.3*	8.3	
2	Push	[ENT] *3.4*	3.4	Y and X values in place.
3	Power	[Y^x]	1333.11	$(8.3)^{3.4}$ obtained.
4	Data	*243.61*	243.61	Number whose root is to be taken.
5	Push	[ENT] *2.7*	2.7	Y and X values in place.
6	Root	[INV] [Y^x]	7.66	Obtain $\sqrt[2.7]{243.61}$
7	Subtract	[−]	1325.45	Compute $(8.3)^{3.4} - \sqrt[2.7]{243.61}$

14. LOGARITHMIC FUNCTIONS

The CORVUS offers four basic logarithmic functions. These functions are common logarithm, natural logarithm (or Lon), powers of ten (10^x), and powers of e (e^x). The following brief review of logarithms should help to clarify the relationship of these four functions.

14.1 Logarithms

The logarithm to the base b of a number x is given by:

if $\log_b x = y$ then $b^y = x$.

Where y is the logarithm, to the base b, of x and x is the antilogarithm of y. Logarithms are useful for multiplication and division and for finding powers and roots. Recall that $b^x \times b^y = b^{x+y}$, $\frac{b^x}{b^y} = b^{x-y}$, $(b^x)^y = b^{xy}$, and $\sqrt[y]{b^x} = b^{x/y}$.

Logarithms provide a means to express problems in these expotential forms. For example, if $a = \log_b c$ (i.e., $c = b^a$) and $d = \log_b f$ (i.e., $f = b^d$) then $cf = b^a \times b^d = b^{a+d}$ and the antilogarithm to the base b of a + d equals cf. Similarly, $c/f \triangleq \text{antilog}_b (a - d)$, $c^f = \text{antilog}_b (fa)$ and $\sqrt[f]{c} = \text{antilog}_b \left(\frac{a}{f}\right)$. Subsection 14.2 will illustrate these applications of the logarithmic functions. Thus with the aid of a table of logarithms, problems involving multiplications and divisions are reduced to additions and subtractions. Problems requiring calculation of powers and roots are simplified to multiplications and divisions. Naturally, the CORVUS makes the multiplication, division, power and root operations extremely easy. However, due to their special properties, logarithms are frequently encountered in a variety of applications.

14.2 Logarithmic Functions on the Corvus

The CORVUS provides a means for direct calculation of logarithms to the two most commonly used bases. Common logarithms (or logarithms to the base 10) are obtained with the keystrokes [DSP] [log]. Natural logarithms (or logarithms to the base e) are obtained with the keystroke [ln]. In both cases the value in the X register is replaced with its logarithm. These operations are only valid for positive values. Either a zero or a negative value in the X register when a logarithm operation is performed will cause the display to flash zeroes.

The antilogarithm of the value in the X register can also be obtained. The keystroke sequence [DSP] [INV] [log] or the sequence [INV] [DSP] [log] will cause the value in the X register to be replaced by its common antilogarithm. The common antilogarithm of a value a is equal to 10^a. The keystrokes [INV] [ln] will calculate the natural antilogarithm of the value in the X register. The natural antilogarithm of a value is equal to e^a.

Antilogarithm operations are defined for positive and negative values. However, these operations can easily result in out of range values. For example, the common antilog of 201 is equal to 10^{201} which is outside the calculator's range.

The following example illustrates the use of logarithms to perform multiplication and division, and to find powers and roots.

Example: $\frac{6.93 \times 5^{4.01}}{\sqrt[3.4]{9.1 \times 10^{30}}} \stackrel{?}{=}$

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Format Display	[DSP] [SCI] [DSP] 9	0.000000000 00	
2	Data	6.93	6.93	
3	log	[DSP] [log]	8.407332346-01	log (6.93)
4	Data	5	5	
5	log	[DSP] [log]	6.989700043-01	log 5
6	Multiply	4.01 [×]	2.802869717 00	log ($5^{4.01}$)
7	Add	[+]	3.643602952 00	log ($6.93 \times 5^{4.01}$)
8	Data	9.1 EE 30	9.1 30	
9	log	[DSP] [log]	3.095904139 01	log (9.1×10^{30})
10	Divide	3.4 [÷]	9.105600410 00	log $\left(\sqrt[3.4]{9.1 \times 10^{30}}\right)$
11	Subtract	[−]	-5.461997458 00	log $\left(\frac{6.93 \times 5^{4.01}}{\sqrt[3.4]{9.1 \times 10^{30}}}\right)$
12	Antilog	[DSP] [INV] [log]	3.451457599-06	$\frac{6.93 \times 5^{4.01}}{\sqrt[3.4]{9.1 \times 10^{30}}}$

The same result (to the 10 digits displayed) is obtained when the calculation is performed without logarithmic functions.

Logarithms are useful in a variety of calculations. One of these is radioactive decay. The following example calculates the half-life of radium. The half-life of a radioactive substance is defined as the time it takes for 50% of the substance to disappear.

Example: If 1% of a quantity of radium disappears in 24 years, then what is the half-life of radium?

The amount present at any time is given by:

$$t = \frac{n \text{Ln} (\frac{A}{A_0})}{\text{Ln}(1-r)}$$

where t = time elapsed
n = units of time
A_0 = initial amount of substance
A = amount at time t
r = percent loss
Ln of course is natural logarithm

For half-life, A = 1/2 A_0 and in this case r = .01. The result will be displayed in scientific notation with 4 digits.

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Format Display	DSP SCI DSP 3	0.000 00	
2	Data	*24*	24	Enter n.
3	ln	ENT .5 ln	-6.931-01	Enter $\ln(\frac{A}{A_0})$ = ln (.5)
4	Multiply	X	-1.664 01	Obtain n $\ln(\frac{A}{A_0})$
5	Data	1	1	Enter constant 1
6	Data	ENT 1 EE CHS 2	1.-02	Enter r
7	Subtract	−	9.900-01	Obtain 1-r
8	ln	ln	-1.005-02	Obtain ln (1-r)
9	Divide	÷	1.655 03	Obtain t

It should also be noted that the keystroke sequence 1 INV ln will leave the value *e* in the X register. The number *e* is, of course, one of the most important and useful mathematical constants. *e* is given by the formula $e = \lim_{Z \to 0} (1+Z)^{1/Z}$.

14.3 Logarithms to any Base

Logarithms to any base can be obtained on the CORVUS using either the natural or the common logarithm functions. The simple derivation of a formula for logarithm to any base and antilogarithm to any base is shown below.

Let $\text{Log}_b X = Y$ Then $b^Y = X$ by definition.

We take the common logarithm of both sides of $b^Y = X$ and obtain $Y \log_{10} b = \log_{10} X$. Thus $Y = \log_b X = \frac{\log_{10} X}{\log_{10} b}$

From $Y \log_{10} b = \log_{10} X$ we obtain $X = \text{antilog}_{10} (Y \log_{10} b)$

The same derivation applies to the natural logarithm function.

Thus $\log_b X = \frac{\log_{10} X}{\log_{10} b} = \frac{\ln X}{\ln b}$

$\text{antilog}_b Y = \text{antilog}_{10} (Y \log_{10} b) = \text{anti ln} (Y \ln b)$

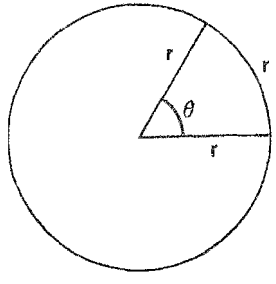
To illustrate the use of logarithms to any base, the following example is presented. Ten digits accuracy can be obtained.

Example: Using logarithms to base 6, find 3^4

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Format	DSP SCI DSP 9	0.000000000 00	
2	ln	*6* ln	1.791759469 00	Find ln 6
3	Store	STO 1	1.791759469 00	Store for future use
4	ln	*3* ln	1.098612289 00	Find ln 3
5	Multiply	*4* X	4.394449155 00	Obtain ln (3^4)
6	Recall	RCL 1	1.791759469 00	Recover ln 6
7	Divide	÷	2.452588771 00	Obtain $\log_6 3^4$
8	Recall	RCL 1	1.791759469 00	Recover ln 6
9	Multiply	X	4.394449155 00	Obtain ln 6 ($\log_6 3^4$)
10	anti ln	INV ln	8.100000000 01	Obtain 3^4

15. ANGULAR UNITS

The CORVUS provides two distinct systems for expressing the measure of an angle – degrees and radians. We are all familiar with degrees as a unit of angle. Each degree represents $\frac{1}{360}$ of the total angle about a point. Radians are also frequently encountered as a unit of angle. Each radian represents the angle subtended by an arc equal to the radius (see figure below)



Circle with radius r

θ = 1 radian = $\frac{360}{2\pi}$ degrees

2π radians = 360°

15.1 Angle Modes

The CORVUS operates in two angular unit modes. When the calculator is first switched on, it is in degree mode. In degree mode all functions with angular measure inputs will interpret those inputs as degrees. All angular measure outputs are given in degrees. The calculator can also be put into radian mode in which all angular measure inputs and outputs are in radians.

The RADIAN key ([RAD]) is utilized to switch between degree mode and radian mode. The calculator can be placed in radian mode with the keystroke sequence [DSP] [RAD] . The calculator will return to degree mode with either [DSP] [INV] [RAD] or [INV] [DSP] [RAD] . The angle mode is indicated by a small dot in the lower right hand corner of the display. When the calculator is in radian mode, the small dot is present. In degree mode, no dot appears.

15.2 Degrees⇌Radians

The CORVUS provides functions to convert between radians and degrees. The TO-RADIAN key ([→RAD]) is utilized to perform these conversions. The keystroke sequence [DSP] [→RAD] will interpret the value in the X register as a degree measure and convert it to radians. A radian value is converted with either [DSP] [INV] [→RAD] or [INV] [DSP] [→RAD] .

The following simple example illustrates conversion between degrees and radians.

Example: 2π radians $\stackrel{?}{=}$ (in degrees)

90 degrees $\stackrel{?}{=}$ (in radians)

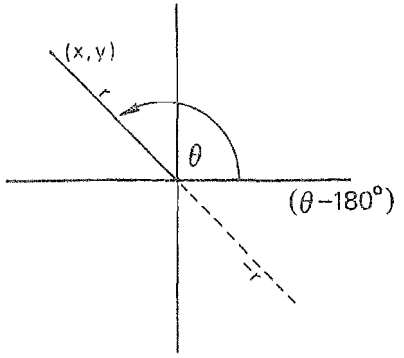
STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENT
1	Data	*2*	2	
2	Pi	[ENT] [DSP] [π]	3.14	
3	Multiply	[X]	6.28	Obtain 2π
4	→Degrees	[DSP] [INV] [→RAD]	360.00	Obtain degree value
5	Clear	[CLX]	0.00	
6	Data	*90*	90	
7	→Radians	[DSP] [→RAD]	1.57	Obtain radian value– should be $\pi/2$
8	Pi	[DSP] [π]	3.14	
9	Divide	[÷]	0.50	yes, it was $\pi/2$

16 POLAR⇋RECTANGULAR COORDINATE CONVERSIONS

Polar coordinates provide a more natural means for specifying points in a plane than the more familiar "x,y" or rectangular coordinate system. Thus polar coordinates are frequently a more useful way of describing points in a plane. The CORVUS offers operations to convert between polar and rectangular coordinates in two dimensions.

16.1 Polar Coordinate Basics

Given a point described by the rectangular coordinates (x,y), we can specify the same point by the polar coordinates (r,θ). The value r is the distance of the point from the origin (i.e., $r = \sqrt{x^2 + y^2}$). θ is the angle between the x axis and the segment connecting (x,y) with the origin (i.e., $\theta = \tan^{-1}\frac{y}{x}$) as in figure 16-1 below.



$(x,y) = (r,\theta)$
$= (r,\theta + 360°)$
$= (-r,\theta + 180°)$
$= (-r,\theta + 180° + 360n°)$
$= (-r,\theta - 180°)$
$= (-r,\theta - 180° - 360n°)$
$= (r,\theta - 360n°)$

Figure 16-1

16.2 Conversion Operations

The CORVUS will convert from rectangular to polar coordinates with the keystroke sequence [DSP] [→POL] . The y coordinate is in the Y register and the x coordinate is in the X register before the conversion. After the conversion, r is in the X register and θ is in the Y register. θ is expressed in either degrees or radians as defined by the calculator mode. The value of θ falls between 0° and 180° (0 and π radians) for positive y values and between 0° and -180° (0 and $-\pi$ radians) for negative y values. This operation will not accept $x = 0$, $y = 0$. Furthermore, some conversions may cause out of range results.

The CORVUS will convert from polar to rectangular coordinates with the keystroke sequence [DSP] [INV] [→POL] or with the sequence [INV] [DSP] [→POL] . The r value is assumed to be in the X register and θ in the Y register. After the conversion operation, the x coordinate is in the X register and the y coordinate is in the Y register. This operation is defined for all values for r and θ, but as θ becomes large the accuracy of the operation gradually decreases. Even at 100,000 revolutions (i.e., 3.6×10^7 degrees), six digits accuracy is maintained.

Example: To illustrate the use of polar - rectangular conversions, the point (5, 29°) is converted to rectangular coordinates and then back to polar coordinates. We will round the calculations to 9 places.

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Format Display	[DSP] [SCI] [DSP] 9	0.000000000 00.	
2	Data	*29*	29.	$29° = \theta$
3	Data	[ENT] *5*	5.	$5 = r$
4	→Rectangular	[DSP] [INV] [→POL]	4.373098536 00	Display x coordinate
5	Radians	[DSP] [RAD]	4.373098536 00.	We will convert back to Polar in radian mode
6	→Polar	[DSP] [→POL]	5.000000000 00.	Same r
7	Exchange	[Y⇔X]	5.061454831-01.	θ in radians
8	→Degrees	[DSP] [INV] [→RAD]	2.900000000 01.	θ in degrees—same θ

16.3 Rectangular⇋Spherical Conversions

The CORVUS can make use of its rectangular⇋polar conversion capability, to convert between rectangular coordinates in three dimensions (x,y,z) and spherical coordinates (r,θ,ϕ). Figure 16-2 below illustrates spherical coordinates.

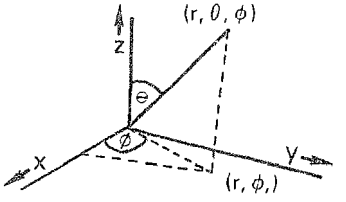


Figure 16 - 2

To convert from rectangular to spherical coordinates, the polar coordinates for the projection of the point into the x-y plane are determined (i.e., $(x,y) \rightarrow (r_0,\theta)$). The angle is ϕ coordinate in spherical coordinates. The polar coordinates for the point (r_0,z) will be equivalent to (r,θ) for the desired point (x,y,z).

Example: Given the point (5, 12, 25) find its spherical coordinates.

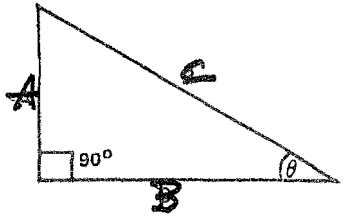
STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Data	12	12	Enter y coordinate
2	Data	[ENT] 5	5	Enter x coordinate
3	→Polar	[DSP] [→POL]	13.00	$(x,y) \rightarrow (r,\varphi)$
4	Data	25	25	Enter z coordinate
5	Exchange	[Y⇔X]	13.00	
6	→Polar	[DSP] [→POL]	28.18	r is displayed; $(r,z) \rightarrow (r,\theta)$
7	Roll	[R↓]	62.53	(r,θ,ϕ) on stack
8	Roll	[R↓]	67.38	Display ϕ

17 TRIGONOMETRIC FUNCTIONS

Trigonometric functions are useful in describing the periodic nature of various phenomena such as pendulums and waves. Trigonometric functions are therefore encountered in many areas of scientific and engineering endeavour. The CORVUS provides operations for calculating the trig functions.

17.1 Trigonometric Function Basics

The trig functions can be defined in terms of the ratio of the sides of a right triangle:



$c = \sqrt{a^2 + b^2}$

sine θ (written sin θ) = a/c

cosine θ (written cos θ) = b/c

tangent θ (written tan θ) = a/b

cosecant θ (written csc θ) = $1/\sin\theta$ = c/a

secant θ (written sec θ) = $1/\cos\theta$ = c/b

cotangent θ (written cot θ) = $1/\tan$ = b/a

Figure 17

Inverse trig functions are similarly defined:

sin -1 a/c = θ	csc -1 c/a = θ
cos -1 b/c = θ	sec -1 c/b = θ
tan -1 a/b = θ	cot -1 b/a = θ

These definitions are only adequate for $0° \leq \theta \leq 90°$. Naturally the trig functions are defined for angles greater than 90° and less than 0°. These definitions are obtained by considering a point (x,y) with polar coordinates (r,θ) with $r \geq 0$. The triangle is formed by the segments connecting (x,y), (x,0) and (0,0). Thus a becomes y, b becomes x, c becomes r, and θ becomes the coordinate θ. The following table summarizes the trig function values by quadrant in the x-y plane.

	I	II	III	IV
	$0°\rightarrow90°$	$90°\rightarrow180°$	$180°\rightarrow270°$	$270°\rightarrow360°$
sin θ	sin θ $0\rightarrow1$	sin $(180°-\theta)$ $1\rightarrow0$	$-\sin(\theta-180°)$ $0\rightarrow-1$	$-\sin(360°-\theta)$ $-1\rightarrow0$
cos θ	cos θ $1\rightarrow0$	$-\cos(180°-\theta)$ $0\rightarrow-1$	$-\cos(\theta-180°)$ $-1\rightarrow0$	cos $(360°-\theta)$ $0\rightarrow1$
tan θ	tan θ $0\rightarrow\infty$	$-\tan(180°-\theta)$ $-\infty\rightarrow0$	tan $(\theta-180°)$ $0\rightarrow\infty$	$-\tan(360°-\theta)$ $-\infty\rightarrow0$
csc θ	csc θ $\infty\rightarrow1$	csc $(180°-\theta)$ $1\rightarrow\infty$	$-\csc(\theta-180°)$ $-\infty\rightarrow-1$	$-\csc(360°-\theta)$ $-1\rightarrow-\infty$
sec θ	sec θ $1\rightarrow\infty$	sec $(180°-\theta)$ $-\infty\rightarrow-1$	sec $(\theta-180°)$ $-1\rightarrow-\infty$	sec $(360°-\theta)$ $\infty\rightarrow1$
cot θ	cot θ $\infty\rightarrow1$	cot $(180°-\theta)$ $0\rightarrow-\infty$	cot $(\theta-180°)$ $\infty\rightarrow0$	cot $(360°-\theta)$ $0\rightarrow-\infty$

17.2 Trigonometric Functions on the CORVUS

The CORVUS provides direct calculation of sin x, cos x, and tan x where the angle x may be expressed in degrees or radians. These functions are obtained with the keystrokes [SIN], [COS], and [TAN] respectively. The value in the X register is interpreted as an angle and is replaced by sin x, cos x, or tan x as is appropriate. The angular mode determines whether the angle is interpreted as degrees or radians. The values csc x, sec x, and cot x can be obtained by following [SIN], [COS], or [TAN] respectively by [DSP] [1/x].

If the angle entered has a tangent of $\pm\infty$ (e.g., tan 90° and tan 270°), the tangent operation will cause the display to flash ±10.00 99 when the display is rounded and 9.999999999 99 when the display is not rounded. Otherwise all operand values are acceptable for all three operations. Accuracy does gradually degrade with the size of the angle. Even at 100,000 revolutions (e.g., 3.60×10^7 degrees), however, 6 digit accuracy is still maintained.

Example: Verify that the fundamental trig identity $\sin^2x + \cos^2x = 1$ holds for 29°. For this problem, ten digit accuracy is desired.

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Format Display	[DSP] [SCI] [DSP] 9	0.000000000 00	
2	Data	*29*	29	Enter angle
3	Push	[ENT]	2.900000000 01	Copy angle into Y register for convenience
4	Sin	[SIN]	4.848096202 −01	Obtain sin (29°)
5	X^2	[DSP] [INV] [$\sqrt{X}$]	2.350403679 −01	Obtain $\sin^2$ (29°)
6	Exchange	[Y⇔X]	2.900000000 01	Recover angle; put $\sin^2$ (29°) in Y register
7	Cos	[COS]	8.746197071 −01	Obtain cos (29°)
8	X^2	[DSP] [INV] [$\sqrt{X}$]	7.649596321 −01	Obtain $\cos^2$ (29°)
9	Add	[+]	1.000000000 00	Obtain $\cos^2$ (29°) + $\sin^2$ (29°)

17.3 Inverse Trigonometric Functions on the CORVUS

The CORVUS provides for direct calculation of $\sin^{-1}x$, $\cos^{-1}x$ and $\tan^{-1}x$. These functions are obtained by depressing the INVERSE key (INV) followed by SIN , COS , or TAN respectively. In these operations, the content of the X register is replaced by an angle expressed in the units dictated by the angular mode. The values $\csc^{-1}x$, $\sec^{-1}x$ and $\cot^{-1}x$ can be obtained by pressing DSP 1/x and then finding $\sin^{-1}x$, $\cos^{-1}x$ or $\tan^{-1}x$, as appropriate.

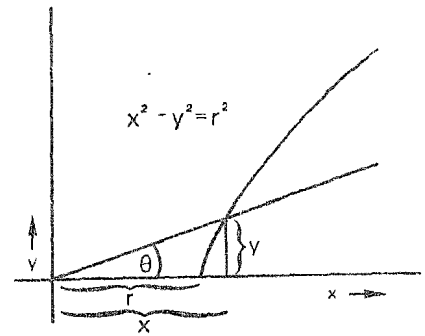
Sin x and cos x always fall in the range -1 to +1. Thus $\sin^{-1}x$ and $\cos^{-1}x$ are undefined if $x>1$ or $x<-1$. If the value in the X register is not in the range -1 to +1, the keystroke sequence INV SIN or INV COS will cause the calculator to flash zeroes.

Example: Given a right triangle with legs of length 5 and 13 find the size of the two acute angles in radians. (8 digits accuracy desired)

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Format Display	DSP SCI DSP 7	0.0000000 00	
2	Radians	DSP RAD	0.0000000 00.	We want angles in radians
3	Data	*5*	5 .	Length of side a
4	Data	ENT *13*	13 .	Length of side b
5	Divide	÷	3.8461538 -01.	Obtain a/b = tan θ
6	Push	ENT	3.8461538 -01.	Copy a/b
7	$\tan^{-1}x$	INV TAN	3.6717383 -01.	Obtain θ
8	Exchange	Y⇔X	3.8461538 -01.	Reattain a/b = cot ϕ
9	Reciprocal	DSP 1/x	2.6000000 00.	Obtain b/a = tan ϕ
10	$\tan^{-1}x$	INV TAN	1.2036225 00.	Obtain ϕ

18 HYPERBOLIC POLAR⇔RECTANGULAR COORDINATE CONVERSIONS

The hyperbolic polar coordinate system is a means for describing a point in terms of its position on a rectangular hyperbola about the origin. The figure below depicts a point in polar hyperbolic coordinates.



(x, y) in rectangular system
$x^2 - y^2 = r^2$
(r, θ) in hyperbolic polar system
θ is expressed in hyperbolic radians

Figure 18

note: rectangular hyperbola is given by $x^2 - y^2 = r^2$

.. hyperbolic polar coordinates can only be applied to (x,y) with $|x| > |y|$

In hyperbolic polar coordinates:

$x > 0, y > 0 \Leftrightarrow r > 0, \theta < 0$
$x > 0, y < 0 \Leftrightarrow r > 0, \theta > 0$
$x < 0, y > 0 \Leftrightarrow r < 0, \theta < 0$
$x < 0, y < 0 \Leftrightarrow r < 0, \theta > 0$

The CORVUS provides a direct means for converting from rectangular to hyperbolic polar coordinates. Simply press either DSP HYP or HYP DSP followed by →POL . The content of the X register is interpreted as the x coordinate and the content of the Y register is interpreted as the y coordinate. After the conversion is complete r is in the X register and θ is in the Y register. Unfortunately, the CORVUS will always return a positive value for r. Therefore the sign, which is the same as the sign of the x coordinate must be remembered.

The CORVUS also provides a direct means to convert from hyperbolic polar to rectangular coordinates. The keystroke sequence requires pressing DSP , INV and HYP in any order followed by →POL . The translation described above is reversed. This operation will, as appropriate, convert to (x,y) in all four quadrants. The relationship $|x| > |y|$ is maintained. The example in section 19 will illustrate these conversions.

19 HYPERBOLIC FUNCTIONS

Hyperbolic functions are encountered in a variety of application areas including physics and electrical engineering. The hyperbolic functions represent relations between the coordinates of a point on a rectangular hyperbola (i.e., $x^2-y^2=r^2$). The hyperbolic functions are given by:

hyperbolic sine of u = $\sinh u = \frac{y}{r} = \frac{e^u - e^{-u}}{2} = \frac{1}{\operatorname{csch} u}$

hyperbolic cosine of u = $\cosh u = \frac{x}{r} = \frac{e^u + e^{-u}}{2} = \frac{1}{\operatorname{sech} u}$

hyperbolic tangent of u = $\tanh u = \frac{y}{x} = \frac{e^u - e^{-u}}{e^u + e^{-u}} = \frac{1}{\coth u}$

The u in the formula above is equivalent to θ in the hyperbolic polar coordinate system. θ is therefore said to be expressed in hyperbolic radians ($\theta = \tanh^{-1} y$).

The hyperbolic functions sinh x, cosh x and tanh x can be directly obtained on the CORVUS by pressing [HYP] followed by either [SIN], [COS], or [TAN] respectively. The value in the X register is replaced by the appropriate hyperbolic function. If the value in the X register is of large magnitude the sinh x and cosh x operations will result in the display flashing ±5 and 10^{99} indicating an out-of-range result. Tanh x will be 1 for any large positive x and 1 for any large negative X. The other hyperbolic functions can be obtained with the RECIPROCAL key and the appropriate hyperbolic function.

The inverse hyperbolic functions $\sinh^{-1}x$, $\cosh^{-1}x$, and $\tanh^{-1}x$ can also be directly calculated on the CORVUS. The inverse hyperbolic functions are obtained by pressing either [INV] [HYP] or [HYP] [INV] followed by either [SIN], [COS] or [TAN]

$\tanh^{-1}x$ is not defined for $x \geq 1$ or $x \leq -1$ and the display will flash zeroes for such operands. The inverse hyperbolic functions replace the content of the X register with the appropriate value for θ.

Example: Given a point with x,y coordinates (4, 3), find the hyperbolic polar coordinates of the point, $\sinh\theta$ and $\cosh^{-1}X$.

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Data	3 [ENT]	3.00	y cordinate
2	Data	4	4.00	x cordinate
3	→Hyp. Pol.	[DSP] [HYP] [→POL]	2.65	r coordinate
4	Data	4	4	Reenter x coordinate
5	Exchange and divide	[Y⇔X] [÷]	1.51	Obtain x/r
6	$\cosh^{-1}x$	[INV] [HYP] [COS]	0.97	Obtain θ
7	Subtract	[−]	0.00	θ from coordinate conversion is the same hopefully not a surprise.

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
8	Recall	[RCL] [LAST X]	0.97	Recover θ
9	Sinh θ	[HYP] [SIN]	1.13	Display $\sinh\theta$

20 SUMMATION, MEAN, AND STANDARD DEVIATION

The CORVUS provides a set of statistical operations which are invaluable in calculation programs for many application areas. The operations provided are the sum of entries, sum of the squares of entries and number of entries, as well as the arithmetic mean and the standard deviation of data entered. In addition the sum of entries in two dimensions is mechanized.

20.1 Entering Data for Statistical Operations

Whenever the SUMMATION key ([Σ+]) is pressed, a series of calculations and store operations are initiated. These operations correspond to the entering of statistical data.

Pressing [Σ+] causes the value in memory 7 to be incremented by 1. Memory 7 therefore contains the data entries in the current sequence.

Pressing [Σ+] also causes memory 8 to be increased by the square of the value in the X register and causes memory 9 to be increased by the value in the X register. Memory 8 therefore contains the sum of the squares of the X data entries and memory 9 contains the sum of the X data entries.

Additionally, the sum of the values in the Y register is maintained. Each time [Σ+] is pressed a special memory is incremented by the value in the Y register. This special memory is called the Σy memory. It can only be accessed by the keystroke sequence [RCL] [Σ+] as will be described in subsections 20.2 and 20.3. Naturally, the Σy memory can be ignored if not needed.

The CORVUS also provides a means for correcting data entries. The keystroke sequence [INV] [Σ+] causes memory 7 to be decremented by 1, memory 8 to be decreased by the square of the value in the X register, memory 9 to be decreased by the value in the X register and the Σy memory to be decreased by the value in the Y register.

It is important to note that neither [Σ+] nor [INV] [Σ+] has any effect upon the stack. Unlike other functions, no push is left pending. Most significantly, the X and Y values remain unchanged. This "residue" is a particularly convenient feature since two dimensional calculation sequences frequently require the X and Y values for further computation (e.g., this feature greatly simplifies calculation of Σxy).

20.2 Memory Manipulations and Restrictions

The summation operation utilizes memories 7, 8, and 9 (in addition to the Σy memory) as accumulating memories for its various calculations. Since these memories are treated as accumulative, it is essential that they be handled carefully so as not to destroy their validity. The CORVUS does a part of this job automatically. The first time [Σ+] is pressed after the calculator is switched on and after the keystroke sequence [DSP] [CLR], the summation memories are cleared. The clearing occurs only when the summation key is actually punched. With the singular exclusion of [DSP] [CLR] any function may be keyed in without interrupting the accumulation of the summation memories. Thus, even if the summation memories are "stored over", when the summation sequence is continued, the memories will continue to accumulate despite their invalid contents.

20.3 Statistical Operations

20.3.1 Recall Summation

The keystroke sequence [RCL] [Σ+] causes the stack to be pushed twice. The value stored in memory 9 is placed in the X register and the value stored in the Σy memory is placed in the Y register. Thus, after a series of data entries, [RCL] [Σ+] puts Σy in the Y register and Σx in the X register.

20.3.2 Mean and Standard Deviation

The keystroke sequence [DSP] [x̄,s] causes the stack to be pushed twice. The arithmetic mean of the X data entries is placed in the X register and the standard deviation of the X data entries is placed in the Y register. To be more correct, the value placed in the X register is equal to $\frac{\text{value in memory 9}}{\text{value in memory 7}}$

This value should be $\frac{\Sigma x}{n}$ = arithmetic mean. If both memory 7 and memory 9 are "tampered with" the display will flash zeroes.

The standard deviation on the CORVUS uses n-1 weighting and thus is given by $\sqrt{\frac{\Sigma x^2 - n\bar{x}^2}{n-1}}$. Obviously this value can be obtained via calculations with memories 7, 8 and 9.

Example: Given the set of points {(10,3), (5, 4), (6,8), (9,7)} find Σy, Σx, x̄, and the standard deviation of x. Additionally, find the variance of x using n weighting without using the mean and standard deviation function. Variance with n weighting: $\frac{\Sigma x^2 - n\bar{x}^2}{n}$ Assume the calculator was just switched on.

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Data	*3*	3	Enter x_i
2	Data	[ENT] *10*	10	Enter y_i
3	Summations	[Σ+]	10.00	

Repeat steps 1,2 and 3 for all data.
Note that the display column for 1,2 and 3 is valid only for the first point.

4	Recall	RCL [Σ+]	30.00	Display Σx
5	Pop	[CLX] [+]	22.00	Display Σy
6	Mean and S.D.	[DSP] [x̄,s]	7.50	Display x̄
7	Pop	[CLX] [+]	2.38	Display Standard Deviation
8	Recall	[RCL] 8	242.00	Recall Σx^2
9	Recall	[RCL] 7	4.00	Recall n
10	Divide	[÷]	60.50	Σx^2 / n
11	Recall	[RCL] 9	30.00	Recall Σx
12	x^2	[DSP] [INV] [√x]	900.00	$(\Sigma x)^2$
13	Recall	[RCL] 7	4.00	Recall n
14	x^2	[DSP] [INV] [√x]	16.00	n^2
15	Divide	[÷]	56.25	$(\Sigma x)^2 / n^2 = \bar{x}^2$
16	Subtract	[−]	4.25	Variance

PART II: APPLICATION PROBLEMS

21 USING PART II

The purpose of the second part of this book is to help extrapolate your basic knowledge of your calculator's operation into a variety of application areas. Each problem is accompanied by a description of the solution approach. The comments are geared toward that description. Thus the motivation for each step in the solution programs should be apparent.

The sections of Part II are almost completely independent. It is not necessary to slowly work through each section in order. Particular areas of interest can be selected or this entire part of the book can simply be used as a reference – a source of solutions and solution programs to problems as they are encountered. An index into Part II by subject is incorporated into the index at the back of the book. The auxiliary formulas and constants in Appendix C should complement the solution program set for use as a reference.

As in Part I, each solution program is laid out in tabular form with step-by-step comments. The format for the problems is virtually identical to Part I. Variable data items are in italics in the programs of Part II. Furthermore the program steps frequently consist of multiple operations. The basic step-by-step solution format is still maintained though.

22 FINANCIAL APPLICATIONS

Simple interest

The basic formula for compound interest is:

$FV = PV(1 + ni)$ Where FV = Future Value
PV = Present Value
n = number of periods
i = interest per period

What will be the value (FV) of $1200 (PV) invested at 5% (i = .05) simple interest for 6 years?

Stack depth used = 4

STEP	FUNCTION	KEYSTROKES		DISPLAY	COMMENTS
1	Data	*1200*	ENT	1200.00	Enter PV
2	Data	1	ENT	6.00	Enter constant
3	Data	*6*	ENT	10.00	Enter n
4	Multiply	*.05*	X	0.30	Obtain ni
5	Add		+	1.30	(1+ni)
6	Multiply		X	1560.00	FV

How much must be invested (PV) at 6% (i-.06) simple interest to result in $2500(FV) after 10 years (n)?

Stack depth used = 4

STEP	FUNCTION	KEYSTROKES		DISPLAY	COMMENTS
1	Data	*2500*	ENT	2500.00	Enter FV
2	Data	1	ENT	1.00	Enter constant
3	Data	*10*	ENT	10.00	Enter n
4	Mulitply	*.06*	X	0.60	Obtain ni
5	Add		+	1.60	(1+ni)
6	Divide		÷	1562.50	PV (amount invested)

Compound Interest

The basic formula for compound interest is:

$FV = PV(1 + i)^n$ using the same notation as above.

What will be the value of $750 placed in a 5.25% compounded quarterly savings account for 1½ years? Note that although the interest has finer resolution, leaving the display mode at 2 decimal places does not reduce the accuracy. The calculator maintains full internal accuracy at all times. Since the final result we desire is simply a dollars and cents result, the 2 decimal place display is appropriate.

Stack depth used = 4

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Data	*750* [ENT]	7500.00	Enter PV
2	Data	1 [ENT]	1.00	Enter constant
3	Data	*.0525* [ENT]	0.05	Annual interest
4	Divide	*4* [÷]	0.01	Convert to i
5	Add	[+]	1.01	(1+i)
6	Data	*1.5* [ENT]	1.50	Enter years
7	Multiply	*4* [X]	6.00	Convert to n
8	Power	[Y^x]	1.08	$(1+i)^n$
9	Multiply	[X]	811.03	FV

Rewriting the basic formula for compound interest

$$PV = \frac{FV}{(1+i)^n}$$

How much must be placed in a 5% annual interest rate compounded monthly savings account to yield $1500 after 1 year?

Stack depth used = 4

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Data	*1500* [ENT]	1500.00	Enter FV
2	Data	1 [ENT]	1.00	Enter constant
3	Data	*.05* [ENT]	0.05	Enter annual interest
4	Divide	*12* [÷]	4.17-03	Convert to i
5	Add	[+]	1.00	(1+i)
6	Data	1 [ENT]	1.00	Enter years
7	Multiply	12 [X]	12.00	Convert to n
8	Power	[Y^x]	1.05	$(1+i)^n$
9	Divide	[÷]	1426.99	PV (amount saved)

From the basic compound interest formula, we get

$$n = \frac{\ln(\frac{FV}{PV})}{\ln(1+i)}$$

How long does it take to double your money at 6% annual interest rate compounded monthly? We know that $\frac{FV}{PV}$ is 2.

Stack depth used = 4

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	ln	*2* [ln]	0.69	Obtain ln 2
2	Data	1 [ENT]	1.00	Enter constant
3	Data	*.06* [ENT]	0.06	Enter annual interest
4	Divide	12 [÷]	5.00-03	Convert to i
5	Add, ln	[+] [ln]	4.99-03	ln(1+i)
6	Divide	[÷]	138.98	n
7	Divide	12 [÷]	11.58	convert to years

Continuous Compounding

The formula for continuous compounding is:

$FV = PV\,e^{in}$

In the competition for customers, the new savings and loan in town is trying to attract customers away from other institutions by offering continuous compounding of interest. You currently have $500.00 in a 5.25% compounded quarterly savings account. The other savings and loan offers the same interest rate, but compounds continuously. What would be the difference in the two accounts after 5 years?

Stack depth used = 2

First perform the program for compound interest to obtain the result for the quarterly compounding, and store that result in memory 1. (648.98)

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Data	*0525* [ENT]	0.05	Enter interest rate
2	Multiply	5 [X]	0.26	Obtain in
3	e^x	[INV] [ln]	1.30	Inverse ln is e^x
4	Multiply	*500* [X]	650.09	Multiply by PV to obtain FV
5	Recall and subtract	[RCL] 1 [−]	1.11	Difference between two compounding rates

Nominal Rate Converted To Effective Annual Rate

The formula to convert nominal annual rate to effective annual rate (after compounding) is:

$$EAR = (1 + i)^n - 1$$

Where i = rate per period
n = periods per year
EAR = effective annual rate

What is the effective annual rate equivalent to 5% compounded monthly?

Stack depth used = 2

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Clear and set display	CLX DSP 4	0.0000	
2	Data, store, divide	.05 ENT 12 STO 1 ÷	0.0042	Enter annual rate, convert to i
3	Add	1 +	1.0042	1+i
4	Recall and power	RCL 1 Y^x	1.0512	$(1+i)^n$
5	Subtract	1 −	0.0512	5.12%

Add-on Rate Converted To True Annual Percentage Rate (APR)

The following formula provides an approximation for the true annual rate equivalent to add-on rate:

$$APR \cong \frac{600ni}{3(n+1) + [(n-1)\, ni/m]}$$

Where: n = number of payments
m = payments per year
i = add-on interest rate

What is the true APR on a 24 month, 5.5% loan?

Stack depth used = 4

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Clear and set display	CLX DSP 2	0.00	
2	Data and store	24 ENT STO 1	24.00	Enter and save n
3	Add	1 +	25.00	n+1
4	Multiply	3 ×	75.00	3(n+1)
5	Recall and subtract	RCL 1 1 −	23.00	n-1
6	Data, recall, multiply	.055 RCL 1 ×	1.32	ni
7	Store	STO 1	1.32	Save ni
8	Data and divide	12 ÷	0.11	Enter m; obtain ni/m
9	Multiply	×	2.53	Multiply by n-1
10	Add	+	77.53	Obtain denominator
11	Data, recall, multiply	600 RCL 1 ×	792.00	Enter constant; multiply by ni
12	Exchange and divide	X⇔Y ÷	10.22	Approximate APR

Annuity

The basic formula for an annuity is:

$$FV = PV \frac{(1+i)^n - 1}{i}$$

If one saves $50.00 per month in a 5% annual rate compounded monthly savings account, what will be the total amount in the account after 5 years?

Stack depth used = 4

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Data	50 ENT	50.00	Enter PV
2	Data	1 ENT	1.00	Enter constant
3	Data	.05 ENT	0.05	Annual interest
4	Divide and store	12 ÷ STO 1	4.17 - 03	Convert to i, save
5	Add	+	1.00	(1+i)
6	Data	5 ENT	5.00	Enter years
7	Multiply	12 ×	60.00	Convert to n
8	Power	Y^x	1.28	$(1+i)^n$
9	Subtract	1 −	0.28	Subtract constant
10	Multiply	×	14.17	Multiply by PV
11	Recall and divide	RCL 1 ÷	3400.30	FV

Loan Payment

A loan payment may be computed from:

$$PMT = \frac{PV \cdot i}{1 - (1+i)^{-n}}$$

where PV = present value or loan amount
i = interest per period
n = number of periods
PMT = payment per period

What is the monthly payment required to pay off a $4250.00 loan in 48 months at an annual rate of 9.5%?

Stack depth used = 3

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Data	.095 ENT	0.10	Annual rate, display rounded
2	Divide	12 ÷	7.92 - 03	Convert to i
3	Push	ENT ENT	7.92 - 03	Save for step 5*
4	Multiply	4250 ×	33.65	PV·i
5	Exchange	X⇔Y	7.92 - 03	Save PV·i, recall i
6	Add	1 +	1.01	1+i
7	Power	48 CHS Y^x	0.68	$(1+i)^{-n}$
8	Subtract	1 X⇔Y −	0.32	$1-(1+i)^{-n}$
9	Divide	÷	106.77	PMT

* After step 3, i will be in X, Y, and Z registers. In step 4, the X register is overwritten by PV and the multiply will pop the stack, leaving i in the Y register.

Remaining Balance

The formula to compute the balance remaining on a loan is:

$$BAL_j = PMT \frac{[1-(1+i)^{j-n}]}{i}$$

n = total number of payments
PMT = payment per period
i = interest per period
j = current period
BAL_j = balance after j^{th} payment

What is the balance remaining after 32 months on a 48 month, $4250.00 loan at 9.5% annual rate (monthly payments are $106.77)?

Stack depth used = 3

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Clear and set display	CLX DSP 2	0.00	
2	Data and divide	.095 ENT 12 ÷	7.92 - 03	Enter annual rate; convert to i
3	Store	STO 1	7.92 - 03	Save i
4	Add	1 +	1.01	1+i
5	Data and subtract	32 ENT 48 −	- 16.00	j-n
6	Power	Y^x	0.88	$(1 - i)^{j-n}$
7	Data, exchange, subtract	1 Y⇔X −	0.12	Obtain numerator
8	Recall and divide	RCL 1 ÷	14.97	Divide by i
9	Enter and multiply	106.77 ×	1598.63	Enter PMT, multiply to obtain BAL_j

Depreciation — Straight Line Method

The formula for straight line depreciation is:

$D=\frac{PV}{n}$ and $DV_j = PV - jD$ Where PV = present value (initial value less salvage value)
n = number of periods of life of asset
D = depreciation per year
DV_j = value after j periods

A truck has an initial value (less salvage) of $3100.00 and an expected life of 5 years. What is the depreciation per year and what is the depreciated value after 3 years?

Stack depth used = 3

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Data	3100 ENT ENT	3100.00	Enter PV, save for step 3
2	Divide	5 ÷	620.00	D
3	Multiply and subtract	3 × −	1240.00	DV_j = PV - jD

Depreciation — Diminishing Balance Method

The formula for diminishing balance depreciation is:

$$D_j = PV_{j-1}\left(1-\left(\frac{S}{PV_0}\right)^{1/n}\right) \text{ and } PV_j = PV_{j-1} - D_j$$

Where
PV_0 = initial value
S = Salvage value
PV_j = Value at period j
D_j = Depreciation at period j
n = periods of life

Using the example problem above, find the depreciation and value for the first three years, using the diminishing balance method. (assume salvage value is $500, thus PV_0 is $3600.00)

Stack depth used = 4

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Data and push	3600 ENT ENT ENT	3600.00	Fills stack with PV_0
2	Divide	500 Y⇔X −	0.14	S/PV_0
3	Reciprocal and power	5 DSP 1/x Y^x	0.67	$(S/PV_0)^{1/n}$
4	Subtract and store	1 Y⇔X − STO 1	0.33	$1 - (S/PV_0)^{1/n}$ and save
5	Multiply	×	1174.31	D_1 (PV_0 in stack)
6	Subtract	−	2425.69	PV_1 (PV_0 in stack)
7	Push	ENT ENT	2425.69	Save PV_{j-1}*
8	Multiply	RCL 1 ×	791.25	D_j*
9	Subtract	−	1634.44	PV_j*

* Steps 7, 8 and 9 are repeated for each successive period desired. The example shows only determining D_2 and PV_2 from PV_1. After step 9, looping back to step 7 would determine D_3 and PV_3 from PV_2. Each repetition of the loop 7, 8, 9 would determine the values for another period. Thus D_3 = 533.15 and PV_3 = 1101.29.

Depreciation — Sum of Years Digits Method

The formula for Sum of Years Digits depreciation (SOD) is:

$D_j = \frac{2(n-j+1)}{n(n+1)}$ PV and $DV_j = S + (n-k)\,D_j/2$ Where

PV = initial value
S = salvage value
n = periods of life
D_j = depreciation at period j
DV_j = value at period j

Again we use the example problem from above. (PV = 3600, n = 5, S = 500, j = 3)

Stack depth used = 4

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Data and push	*5* ENT ENT	5.00	Enter n, save for step 6
2	Subtract and store	*3* − STO 1	2.00	n − j, save for step 1
3	Add	1 +	3.00	Add constant, (n − j + 1)
4	Multiply	2 X	6.00	2(n − j + 1)
5	Multiply	*3600* X	21600.00	2(n − j + 1)PV
6	Exchange	Y⇔X	5.00	restore n, save 2(n − j + 1)PV for step 1
7	Push	ENT ENT	5.00	save n for step 9*
8	Add	1 +	6.00	Add constant, n+1
9	Multiply	X	30.00	n(n+1)
10	Divide	÷	720.00	D_j
11	Multiply	RCL 1 X	1440.00	$(n-j)D_j$
12	Divide	2 ÷	720.00	$(n-j)D_j/2$
13	Add	*500* +	1220.00	DV_j

* After step 7, n will be in the X, Y and Z registers. The data entry in step 8 will overwrite X and the add will pop the stack, leaving n in the Y register.

23 SERIES AND PROGRESSIONS

Arithmetic Progressions

An arithmetic progression is defined by

$$a,\ a+i,\ a+2i,\ a+3i,\ \ldots\ a+(n-1)i$$

For our example, we will step through an arithmetic progression with a = 10 and i = 7.

Stack depth used = 4*

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Data	*7* ENT ENT ENT	7.00	Fill stack with i
2	Data and add	*10* +	17.00	Enter a ; add to obtain second term
3	Add	+	24.00	Add i to obtain next term

Repeat step 3 for each succeeding term.

The preceeding program fills the stack with i to utilize the automatic copy of the bottom of the stack to insure an unending supply of i's for each step. If there is data on the stack that must not be destroyed, the arithmetic progression can be computed with the following program:

Stack depth used = 2

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Data	*7* STO 1	7.00	Save i
2	Data and add	*10* +	17.00	Enter a ; add to obtain second term
3	Recall and add	RCL 1 +	24.00	Add i to obtain next term

Repeat step 3 for each succeeding term.

The n^{th} term of an arithmetic progression is given by

$$a + (n-1)i$$

To obtain the 14^{th} term of the progression above:

Stack depth used = 3

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Data	*10* ENT	10.00	Enter a
2	Data and subtract	*14* ENT −	13.00	Obtain n − 1
3	Data multiply, add	*7* X +	101.00	n^{th} term

Sum of Arithmetic Progression

The sum of the terms of an arithmetic progression (given a and i) is:

$$SUM = na + \frac{n(n-1)i}{2}$$

Where n = number of terms to be summed
a = first term
i = interval

Compute the sum of the first 12 terms of the progression above.

Stack depth used = 4*

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Data	*12* ENT ENT ENT	12.00	3 n's on stack
2	Subtract	1 −	11.00	n - 1
3	Data and multiply	*7* X	77.00	(n - 1) i
4	Multiply	X	924.00	Multiply by n
5	Divide	2 ÷	462.00	Divide by 2
6	Exchange	Y⇔X	12.00	Bring n to top of stack
7	Data and multiply	*10* X	120.00	Multiply by a
8	Add	+	582.00	Obtain SUM

When only the first and last terms and the number of terms are known, the following formula expresses the sum of an arithmetic progression:

$$SUM = \frac{n}{2}(a+t)$$

Where n = number of terms
a = first term
t = last term

The 14th term of the above progression is 101.00. Find the sum of the first 14 terms.

Stack depth used = 3

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Data and divide	*14* ENT 2 ÷	7.00	n/2
2	Data	*10* ENT	10.00	Enter a
3	Data and add	*101* +	111.00	Add t
4	Multiply	X	777.00	Obtain SUM

Geometric Progression

A geometric progression is defined by

$$a, ar, ar^2, \ldots, ar^{n-1}$$

For the example, we will step through a geometric progression with a = 3 and r = 0.9

Stack depth used = 4*

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Data	*.9* ENT ENT ENT	0.90	Fill stack with r
2	Data and multiply	*3* X	2.70	Enter a ; multiply to obtain second term
3	Multiply	X	2.43	Multiply by r to obtain next term

Repeat step 3 for each succeeding term

As in the arithmetic progression, the program above uses the entire stack to take advantage of the automatic copy into the bottom of the stack. A program which uses only two levels of the stack is:

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Data	.9 STO 1	0.90	Save r
2	Data and multiply	*3* X	2.70	Enter a ; multiply to obtain second term
3	Recall and multiply	RCL 1 X	2.43	Multiply by r to obtain next term

Repeat step 3 for each succeeding term.

The nth term of a geometric progression is defined as

$$ar^{n-1}$$

To obtain the 11th term of the geometric progression above:

Stack depth used = 3

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Data	*3* ENT	3.00	Enter a
2	Data and subtract	*11* ENT 1 −	10.00	Obtain n - 1
3	Data, power, multiply	*.9* Y⇔X Y^x X	1.05	nth term

Sum of Geometric Progression

The sum of a geometric progression to n terms is given by:

$$SUM = \frac{a(r^n-1)}{r-1}$$

Where a = initial term
r = ratio between terms
n = number of terms

Compute the sum of the first 15 terms of the above progression.

Stack depth used = 4

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Data	3 ENT	3.00	Enter a
2	Data	15 ENT	15.00	Enter n
3	Data and store	.9 STO 1	0.90	Store r
	If r is not negative, skip to step 5			
4	Negate	CHS	*	Base for a power must be not be negative
5	Exchange and power	Y⇔X Y^x	0.21	r^n
	If r is not negative, skip to step 7; if r is negative and n is even skip to step 7			
6	Negate	CHS	*	Result of odd negative power is negative
7	Subtract	1 −	-0.79	r^n-1
8	Recall and Subtract	RCL 1 1 −	-0.10	Obtain r-1
9	Divide	÷	7.94	$(r^n-1)/(r-1)$
10	Multiply	X	23.82	Multiply by a to obtain SUM

* No display indicated because example has positive r.

Harmonic Progression

A harmonic progression is defined by

$$\frac{d}{b}, \frac{d}{b+c}, \frac{d}{b+2c}, \ldots, \frac{d}{b+(n-1)c}$$

For our example, we will step through the Harmonic progression with d = 2, b = 3, and c = 5.

Stack depth used = 4

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Data and save	2 STO 1	2.00	Save d
2	Push	5 ENT	5.00	Push c for step 6
3	Data and reciprocal	3 ENT DSP 1/x	0.33	Save b; obtain 1/b
4	Recall and multiply	RCL 1 X	0.67	Obtain d/b, 1st term
5	Pop	CLX +	3.00	Throw away previous term; move denominator to top of stack
6	Add and push	+ ENT	8.00	Add c to previous denominator, save for next cycle; *
7	Reciprocal	DSP 1/x	0.13	Reciprocal of denominator
8	Recall and multiply	RCL 1 X	0.25	multiply by a to obtain next term

Repeat steps 5, 6, 7, and 8 for each succeeding term.

* c was entered in step 2. Subsequent operations pushed it to the bottom of the stack. Automatic copy of the bottom of the stack on a pop insures that c will always be on the stack.

To obtain the n^{th} term of a harmonic progression, we note that the general term for a harmonic progression is simply d multiplied by the reciprocal of the n^{th} term of an arithmetic progression (a + (n-1)i; where a = b and i = c). To calculate the 5^{th} term of the example harmonic progression:

Stack depth used = 3

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Data	3 ENT	3.00	a (a=b) *
2	Data and subtract	5 ENT 1 −	4.00	Obtain n-1 *
3	Data, multiply, add	5 X +	23.00	n^{th} term (i=c) *
4	Reciprocal	DSP 1/x	0.04	Obtain denominator
5	Data and multiply	2 X	0.09	Multiply by d to obtain n^{th} term

* Steps 1, 2, and 3 are the program to calculate n^{th} term of an arithmetic progression.

Fibonacci Series

A Fibonacci series is defined by the following relation

$f_i = f_{i-1} + f_{i-2}$ where f_i is the i^{th} term

That is, each term is the sum of the two previous terms.

For our example, we will step through the sequence that begins with $f_1 = 1$ and $f_2 = 1$

Stack depth used = 3*

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Data and save	1 STO 1	1.00	Save f_1
2	Data and push	1 ENT	1.00	Push f_2
3	Exchange memory	STO RCL 1	1.00	Move f_{i-2} onto stack save f_{i-1} for next term
4	Add and push	+ ENT	2.00	Obtain f_1 ; push f_1 for next cycle

Repeat steps 3 and 4 for each succeeding term.
(f_i becomes f_{i-1} and f_{i-1} becomes f_{i-2})

Note: *Stack depth used can be reduced to 2 if CLX is performed between step 1 and step 2.

The formula for n^{th} Fibonacci number is:

$$F_n = \frac{[1/2\ (1+\sqrt{5})]^n}{\sqrt{5}}$$

For our example, compute the 6^{th} Fibonacci number.

Stack depth used = 2

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Clear and set display	CLX DSP 0	0.	
2	Square root and store	5 DSP √x STO 1	2.	Save $\sqrt{5}$
3	Add	1 +	3.	$(1+\sqrt{5})$
4	Divide	2 ÷	2.	$(\frac{1}{2}\ (1+\sqrt{5})$
5	Data and power	*6* Y^x	18.	$(\frac{1}{2}\ (1+\sqrt{5}))^n$
6	Recall and divide	RCL 1 ÷	8.	Divide by $\sqrt{5}$

24 PROBABILITY AND STATISTICS

Means

The CORVUS provides direct calculation of arithmetic means. Two other means can be easily obtained.

Geometric Mean

The geometric mean of a series $(a_1, a_2, a_3, \ldots, a_n)$ is defined as:

Geometric Mean $= \sqrt[n]{a_1 \cdot a_2 \cdot a_3 \cdot \cdots \cdot a_n}$

The program below calculates the geometric mean. For this example the series (5,10,3,6,9) is utilized.

Stack depth used = 2

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Data	*5* ENT	5.00	Enter a_1
2	Data	*10*	10	Enter next a_1
3	Multiply	X	50.00	Obtain $a_1 : : : a_i$
	Repeat steps 2 and 3 for all a_i Note: the keystrokes and display for steps 2 and 3 apply to a_2 only			
4	Data	*5*	5	Enter n
5	$\sqrt[x]{Y}$	INV Y^x	6.05	Display geometric mean

Harmonic Mean

The harmonic mean of a series $(a_1, a_2, \cdots, a_n)$ is defined as:

Harmonic Mean $= \frac{n}{\sum_{i=1}^{n}(1/a_i)}$. The program below calculates the harmonic mean. The same series (5, 10, 3, 6, 9) is utilized in this example.

Stack depth used = 3

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Clear stack	DSP CLR	0.00	Reset statistical memori clear entire stack
2	Data	*5*	5	Enter a_i
3	Reciprocal	DSP 1/x	0.20	Obtain $1/a_i$
4	Σ	Σ+	0.20	Obtain $\sum_{j=1}^{i} 1/a_i$
	Repeat steps 2, 3 and 4 for entire set of values note: the keystrokes and display for steps 2, 3 and 4 apply to a_i only			
5	Recall	RCL 7	5.00	Recall n
6	Recall	RCL 9	0.91	Recall $\Sigma\ 1/a_i$
7	Divide	÷	5.49	Obtain harmonic mean

Permutations and Combinations

The set of permutations of n things taken k at a time is all the ways we can pick an ordered sample of size k from n. For example, if five cards are pulled from a deck one at a time, how many sequences can be drawn? The first card may be any of 52, the second any of 51, etc. Thus the number of permutations in this case is 52 • 51 • 50 • 49 • 48. In general the number of permutations of n things taken k at a time equals $\frac{n!}{(n-k)!}$

The set of combinations of n things taken k at a time is all the possible sets of size k that can be selected from n things. The sample is no longer ordered. Thus the five cards selected could have been selected in any order. From the formula for permutations above, the 5 cards taken 5 at a time can be permuted in $\frac{5!}{(5-5)!}$ ways or 5! ways. In general, the number of combinations of n things taken k at a time (written $\binom{n}{k}$) equals $\frac{n!}{k!(n-k)!}$

Example: Using the formula for permutations $-\frac{n!}{(n-k)!}$ find the number of 5 card sequences which can be dealt from a 52 card deck.

Stack depth used = 3

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Data	*52*	52	Enter n
2	Factorial	DSP X!	8.07 67	Find n!
3	Recall	RCL LAST X	52.00	Recover n
4	Data	*5*	5	Enter k
5	Subtract	−	47.00	Obtain (n−k)
6	Factorial and Divide	DSP X! ÷	311875200.00	Obtain $\frac{n!}{(n-k)!}$

Example: Using the formula for combinations, $\frac{n!}{k!(n-k)!}$ find the number of 5 card poker hands.

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Data	*52*	52	Enter n
2	Factorial	DSP X!	8.07 67	Find n!
3	Recall	RCL LAST X	52.00	Recover n
4	Data	*5*	5	Enter k
5	Subtract	−	47.00	Obtain n−k
6	Recall	RCL LAST X	5.00	Recover k
7	Factorial	DSP X!	120.00	Obtain k!
8	Exchange and Factorial	Y⇔X DSP X!	2.59 59	Obtain (n−k)!
9	Multiply and divide	X ÷	2598960.00	Obtain $\binom{52}{5}$

Permutations and Combinations are very useful in simple probability calculations.

Example: What is the probability of receiving a bridge hand of 13 cards in which no card is higher than a 9 and in which there are 4 spades and 3 cards in each of the other suits?

We know that the probability of choosing a sample with a particular composition is given by $\frac{\binom{r_1}{k_1}\binom{r_2}{k_2}\cdots\binom{r_m}{k_m}}{\binom{r}{n}}$ where $\binom{r}{n}$ represents the total available sample space and the $\binom{r_c}{k_c}$'s represent the composition specifications.

Probability of hand = $\frac{\binom{8}{4}\binom{8}{3}\binom{8}{3}\binom{8}{3}}{\binom{52}{13}}$ (spades)

Stack depth used = 4

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Data and Factorial	*8* DSP X!	40320.00	Start calculating first part composite—8!
2	Data and Factorial	*4* DSP X!	24.00	4!
3	x^2	DSP INV $\sqrt{x}$	576.00	4! 4!
4	Divide	÷	70.00	Obtain $\binom{8}{4}$
5	Data and Factorial	*8* DSP X!	40320.00	8!
6	Data and Factorial	*3* DSP X!	6.00	3!
7	Data and Factorial	*5* DSP X!	120.00	5!
8	Multiply	X	720.00	3! 5!
9	Divide	÷	56.00	$\binom{8}{3}$
10	y^x	*3* Y^x	175616.00	$\binom{8}{3}\binom{8}{3}\binom{8}{3}\binom{8}{4}$
11	Multiply	X	12293120.00	$\binom{8}{3}\binom{8}{3}\binom{8}{3}\binom{8}{3}$
12	Data and Factorial	*52* DSP X!	8.07 67	52!
13	Data and Factorial	*13* DSP X!	6227020800.00	13!
14	Data and Factorial	*39* DSP X!	2.04 46	39!
15	Multiply and Divide	X ÷	635013559596.	$\binom{52}{13}$
16	Divide	÷	1.94 − 05	Probability of hand

Binomial Distribution

Consider repeated independent trials of an experiment. The outcome of each trial can be considered either a success or a failure. The probability of a success is p – the probability of a failure is q = 1-p. The binomial probability of exactly k successes in n trials is: $\binom{n}{k} p^k q^{n-k}$, where $\binom{n}{k}$ is the binomial coefficient defined as $\frac{n!}{k!\,(n-k)!}$. The probability of no success is q^n and the probability of at least one success is therefore $1-q^n$.

A fair die is tossed 8 times. A success is defined as either a 1 or a 6. Thus p = ⅓ and q = ⅔. What is the probability of no successes?

What is the probability of exactly two successes?

Stack depth used = 4

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Data, reciprocal, store	*3* DSP 1/x STO 1	0.33	Store p
2	Data, divide, store	*2* ENT 3 ÷ STO	0.67	Store q
3	Data, power	*8* Y^x	0.04	Obtain q^n; Prob of no success
4	Clear, recall	CLX RCL LAST X	8.00	Recover n
5	Factorial	DSP X!	40320.00	n!
6	Recall	RCL LAST X	8.00	Recover n
7	Data, subtract, store	*2* – STO 4	6.00	n – k, save n – k
8	Recall, store	RCL LAST X STO 3	2.00	Recover k, save k
9	Factorial	DSP X!	2.00	k!
10	Exchange and Factorial	Y⇔X DSP X!	720.00	(n–k)!
11	Multiply, divide	X ÷	28.00	$\binom{n}{k}$
12	Recall, power	RCL 1 RCL 3 Y^x	0.11	p^k
13	Multiply	X	3.11	$\binom{n}{k} p^k$
14	Recall, power	RCL 2 RCL 4 Y^x	0.09	q^{n-k}
15	Multiply	X	0.27	$\binom{n}{k} p^k q^{n-k}$ = prob of k successes

Note: Steps 4 - 11 compute the combination $\binom{n}{k}$ with some additional data stores as required for the binomial formula.

Hypergeometric Distribution

The Hypergeometric probability function is used for selection without replacement from a population which consists of k elements of one type and n-k elements of another type. When s elements are selected, the probability that exactly x of them are of the first type is given by:

$$\frac{\binom{k}{x}\binom{n-k}{s-x}}{\binom{n}{s}}$$ expressed as factorials this becomes

$$\frac{k!\,(n-k)!\,s!\,(n-s)!}{x!\,(k-x)!\,(s-x)!\,[n-k-(s-x)]!\,n!}$$

An urn contains 10 balls, 3 of which are red. If 5 balls are drawn, what is the probability that exactly 2 are red?

Stack depth used = 4* (uses automatic copy of bottom of stack feature)

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Data, push	*10* ENT ENT ENT	10.00	Fills stack with n
2	Data, subtract, store	*3* – STO 1	7.00	n – k, save
3	Recall, store	RCL LAST X STO 2	3.00	recall k, save
4	Factorial	DSP X!	6.00	k!
5	Exchange, factorial	Y⇔X DSP X!	5040.00	(n–k)!
6	Multiply	X	30240.00	(n–k)!k!
7	Exchange	Y⇔X	10.00	Recover n
8	Data, subtract	*5* –	5.00	n–s
9	Recall, store	RCL LAST X STO 3	5.00	recall s, save
10	Factorial	DSP X!	120.00	s!
11	Exchange, Factorial	Y⇔X DSP X!	120.00	(n–s)!
12	Multiply, multiply	X X	435456000.00	k!(n–k)!s!(n–s)!
13	Exchange, factorial, divide	Y⇔X DSP X! ÷	120.00	k!(n–k)!s!(n–s)!/n!
14	Store	STO 9	120.00	Save
15	Data, push	*2* ENT ENT ENT	2.00	Fill stack with x
16	Recall, exchange subtract	RCL 2 Y⇔X –	1.00	k–x
17	Factorial	DSP X!	1.00	(k–x)!
18	Exchange, recall, exchange	Y⇔X RCL 3 Y⇔X	2.00	Recover s
19	Subtract, store, Factorial	– STO 4 DSP X!	6.00	(s–x)!, save s–x
20	Multiply	X	6.00	(s–x)!(k–x)!
21	Exchange, Factorial, Multiply	Y⇔X DSP X! X	12.00	(s–x)!(k–x)!x!
22	Recall, recall, subtract	RCL 1 RCL 4 –	4.00	n–k–(s–x)
23	Factorial, multiply	DSP X! X	288.00	(n–k–(s–x))!(s–x)!·(k–x)!x!
24	Recall, exchange, divide	RCL 9 Y⇔X ÷	0.42	Probability of exactly 2 red balls

Poisson Distribution

The Poission probabilities are defined as:

$$P(x) = \frac{(zt)^x e^{-zt}}{x!}, \quad x = 0, 1, 2, \ldots$$

A common interpretation is that z = mean rate of occurrence of some event, t = time interval, then P(x) is the probability of exactly x events in time interval t. Customers arrive at a store at a rate of 45 per hour. What is the probability that there are no customers arriving in a five minute period?
(When $x = 0$, $P(x) = e^{-zt}$)

Stack depth = 2

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Data	*45* [ENT]	45.00	Rate per hour
2	Data, divide	60 [÷]	0.75	Rate per minute
3	Data, multiply	*5* [X]	3.75	Obtain zt
4	Push, negate	[ENT] [CHS]	-3.75	Save zt for step 6; obtain -zt
5	e^x, store	[INV] [ln] [STO] 1	0.02	Probability of no customers for 5 min. Save for step 7

If there is one clerk, what is the probability that in a five minute period more customers will arrive than can be served? Note that the probability of more than 1 customer arriving is 1-P(1)-P(0). Also note that $P(x=n) = \frac{P(x=n-1)\ zt}{n}$

Assume P(0) remains in X from previous problem.

Stack depth used = 2

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
6	Multiply	[X]	0.09	zt P(0) = P(1)
7	Recall, add	[RCL] 1 [+]	0.11	P(0) + P(1)
8	Data, exchange, subtract	*1* [Y⇔X] [−]	0.89	Probability of more than one customer

Normal Curve

The standard normal curve is defined by:

$$\phi(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}$$

which has mean = 0 and variance = 1. x is in normalized units.

This defines the probability density function. The associated distribution function

$$\frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-x^2/2}\, dx$$

has no simple expression.

What is the value of $\phi(1.7)$?

Stack depth used = 4

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Data	*1.7* [ENT]	1.70	Enter x
2	Square	[DSP] [INV] [$\sqrt{x}$]	2.89	x^2
3	Divide, negate	2 [÷] [CHS]	-1.45	$-x^2/2$
4	e^x	[INV] [ln]	0.24	$e^{-x^2/2}$
5	Pi, multiply, root	[DSP] [π] 2 [X] [$\sqrt{x}$]	2.51	$\sqrt{2\pi}$
6	Reciprocal	[DSP] [1/x]	0.40	1/2
7	Multiply	[X]	0.09	Obtain $\phi(x)$

Chi-Square Statistics

A statistic which is the sum of squares of independent standard normal random variables is said to be Chi-square distributed. There are infinitely many Chi-square distributions, one corresponding to each positive integer, called the degree of freedom. The density function for a Chi-square is quite imposing, and can be obtained from tables. The Chi-square is used in statistical inference about population variances. If s is the observed sample variance of a sample size n taken from a normal population with expected variance v, $\frac{(n-1)\,s}{v}$ Chi-square with n-1 degrees of freedom.

A manufacturing process for light bulbs is sampled and tested for bulb life. A sample of 20 bulbs has an acceptable sample mean, but the sample variance is 300 hours. Specifications require a population variance of 250 hours or less. Is this sample sufficient evidence to reject the lot of bulbs? The hypothesis being tested is that the population variance is 250 or less. The test statistic is Chi-square $= \frac{(n-1)\,s}{v}$; where $s = 300$, $v = 250$, $n = 20$, degree of freedom $= 19$. The level of significance is arbitrarily set as 0.05.

From a table of Chi-square values, we obtain the probability of a Chi-square of 19 degrees of freedom being greater than 30.14 is 0.05. Thus we accept the hypothesis (and the lot of bulbs) if the observed Chi-square is less than 30.14. We reject the hypothesis if the observed Chi-square is greater than or equal to 30.14.

Stack depth used = 2

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Data	*20* [ENT]	20.00	Enter n
2	Subtract	1 [−]	19.00	Obtain n−1
3	Data, multiply	*300* [×]	5700.00	(n−1) s
4	Data, divide	*250* [÷]	22.80	(n−1) s / v

Since the observed Chi-square is less than 30.14 the hypothesis is accepted, and this sample is not sufficient evidence to reject the lot of bulbs.

Least Squares Linear Regression

A least squares linear regression is a mechanism to find a "straight line of best fit" between a pair of independent variables (i.e., variables for which no known dependence exists). Basically, this approach attempts to minimize the sum of the squares of the deviations from a straight line.

The line is given by: $y = mx + b$

where $m = \frac{n\Sigma xy - \Sigma x \Sigma y}{n\Sigma x^2 - (\Sigma x)^2}$ and $b = \overline{y} - m\overline{x}$.

A primary output of a linear regression is a measure of the dependence of the two variables. The correlation coefficient is given by:

correlation coefficient = $r = m\frac{\sigma_x}{\sigma_y}$

with n weighting – $\sigma_x = \sqrt{\frac{\Sigma x^2 - nx^2}{n}}$ and $\sigma_y = \sqrt{\frac{\Sigma y^2 - ny^2}{n}}$

with (n-1) weighting – $\sigma_x = \sqrt{\frac{\Sigma x^2 - nx^2}{n-1}}$ and $\sigma_y = \sqrt{\frac{\Sigma y^2 - ny^2}{n-1}}$

The weighting does not matter for calculating r as long as we are consistent.

The program below calculates the slope and intercept of the least squares best fit and the correlation coefficient. Two special features of the CORVUS are utilized; exchange with memory and the unaltered x,y registers after [Σ+] The program is valid for an arbitrary number of (x,y) pairs.

Example: find the linear least squares fit for:

x	y
5.01	6.52
9.98	11.34
15.21	17.48
19.88	15.08
24.98	18.30

Display is assumed to be set to [DSP] [INV] [SCI] [DSP] 2.

In the program below the first two steps are preparatory and only necessary if the stack and memory 1 have been used since the calculator was switched on. Steps 3-13 are performed for each x,y pair. This part of the program is straight forward and relatively short. Steps 14-37 are only executed once for each least squares regression.

Stack depth used = 3

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Clear	DSP CLR	0.00	Clear stack; reset memories
2	Store	STO 1	0.00	Clear memory 1
3	Data	*6.52*	6.52	Enter y_i
4	Push	ENT	6.52	Copy y_i
5	Multiply	X	42.51	Obtain y_i^2 for step 13
6	Recall	RCL LAST X	6.52	Recall y_i
7	Data	*5.01*	5.01	Enter x_i
8	Summation	Σ+	5.01	Add (x, y) into total
9	Multiply	X	32.67	Obtain $x_i y_i$
10	Recall	RCL 1	0.00	Recall $\Sigma x_i y_i$
11	Add and Store	+ STO 1	32.67	Store $\Sigma x_i y_i$
12	Pop	CLX +	42.51	y_i^2 on X register
13	Add	+	42.51	find Σy_i^2

Repeat steps 3 - 13 for all x,y pairs.

note: the values displayed and data entered in steps 3 - 13 are for the first x,y pair only.

14	Store	STO 2	1038.95	Store Σy^2
15	Recall	RCL 7	5.00	Recall n
16	Recall and Multiply	RCL 1 X	5843.17	Obtain $n\Sigma xy$
17	Recall and Multiply	RCL Σ+ X	5158.12	Obtain $\Sigma x \Sigma y$
18	Subtract	−	685.04	Obtain $n\Sigma xy - \Sigma x \Sigma y$
19	Recall	RCL 7	5.00	Recall n
20	Recall and Multiply	RCL 8 X	6876.30	Obtain $n\Sigma x^2$
21	Recall and x^2	RCL 9 DSP INV $\sqrt{x}$	5634.00	Obtain $(\Sigma x)^2$
22	Subtract and Divide	− ÷	0.55	Obtain m
23	Store	STO 3	0.55	Store m
24	Recall	RCL Σ+	75.06	Obtain Σx, $\Sigma y = n\bar{x}$, $n\bar{y}$
25	Recall, Mult. and subtract	RCL 3 X −	27.33	Obtain $n(\bar{y} - m\bar{x})$
26	Recall and divide	RCL 7 ÷	5.47	Obtain b

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
27	Store	STO 4	5.47	Store b
28	Recall	RCL 2	1038.95	Recover y^2
29	Recall	RCL Σ+ CLX +	68.72	Recall Σx, Σy -- pop Σx
30	x^2, Recall and Divide	DSP INV $\sqrt{x}$ RCL 7 ÷	944.49	Obtain $n\bar{y}^2$
31	Subtract	−	94.47	Obtain $\Sigma y^2 - n\bar{y}^2$
32	Recall and Subtract	RCL 7 1 −	4.00	Obtain n-1
33	Divide and $\sqrt{x}$	÷ DSP $\sqrt{x}$	4.86	Obtain σ_y with n-1 weighting
34	S. D.	DSP $\bar{x}$,s CLX +	7.88	Obtain $\bar{x}$, S. D. pop $\bar{x}$
35	Recall and Multiply	RCL 3 X	4.35	Obtain $m\sigma x$
36	Exchange and Divide	Y⇔X ÷	0.89	Obtain r
37	Store	STO 5	0.89	Store r

at the end, m is in memory 3
b is in memory 4
r is in memory 5

This same program can be used for other types of correlation. When a data item is first entered, any of a number of operations can be performed on the data items. A logarithmic or semi-logarithmic curve fit can be obtained by taking the logarithm of both or one of the variables respectively. Other functions which might be used include trigonometric, hyperbolic, powers and roots.

25 NUMERICAL METHODS

Quadratic Equation

The roots of the quadratic equation $Ax^2 + Bx + C = 0$ are:

$\frac{-B \pm \sqrt{B^2-4AC}}{2A}$ if $D = (B^2-4AC)/4A^2$ is positive, the roots are real.

if D is negative, the roots are complex and expressed as: $\frac{-B}{2A} \pm \frac{\sqrt{4AC-B^2}}{2A}\, j$

That is, $\frac{-B}{2A}$ is the real component and $\frac{\sqrt{4AC-B^2}}{2A}$ is the imaginary component.

Solve $3x^2 + 6x + 4 = 0$ Stack depth used = 4

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Data and Enter	3 [ENT] [ENT]	3.00	Enter A and push
2	Multiply	4 [X]	12.00	Obtain 4A; A still on stack
3	Data and store	6 [CHS] [STO] 1	-6.00	Enter and store -B
4	Square	[DSP] [INV] [√x]	36.00	B^2
5	Exchange	[Y⇔X]	12.00	bring back 4A
6	Data and multiply	4 [X]	48.00	Enter C, obtain 4AC
7	Subtract	[−]	-12.00	B^2-4AC
8	Exchange	[Y⇔X]	3.00	Bring back A
9	Square	[DSP] [INV] [√x]	9.00	A^2
10	Multiply	4 [X]	36.00	$4A^2$
11	Divide	[÷]	-0.33	obtain D
	Skip to step 13 if D not negative			
12	Negate	[CHS]	0.33	Make D positive for √
13	Square root, save	[DSP] [√x] [STO] 2	0.58	Imaginary component of complex root, if D neg.
14	Exchange	[Y⇔X]	3.00	Bring up A
15	Multiply	2 [X]	6.00	2A
16	Recall	[RCL] 1	-6.00	-B
17	Exchange, divide	[Y⇔X] [÷]	-1.00	Obtain -B/2A, real component if D neg
	If D was negative, done			
18	Store	[STO] 1	*	Store -B/2A
19	Add	[+]	*	$\frac{-B+\sqrt{B^2-4AC}}{2A}$
20	Recall	[RCL] 1 [RCL] 2	*	Recall -B/2A, $\frac{\sqrt{B^2-4AC}}{2A}$
21	Subtract	[−]	*	$\frac{-B-\sqrt{B^2-4AC}}{2A}$

* Example had no real roots

Roots of Polynomials

The Newton-Raphson method may be used to compute a root of a polynomial equation: $f(x) = a_0x^n + a_1x^{n-1} + \ldots + a_{n-1}x + a_n = 0$

The Newton-Raphson method is an iterative approximation method described by: $x_{i+1} = x_i - \frac{f(x)}{f'(x)}$ where f'(x) is the derivative of f(x).

Recall that the derivative of a sum is the sum of the derivatives, and that the derivative of a general term ax^n is anx^{n-1}.

For the example, approximate a root of $f(x) = 2x^3-5x^2+35x-15$. Use the initial value $x_0 = 1$. The program given here is applicable to a polynomial with 7 or less coefficients. Coefficients are stored in memories 1-7, while f(x) and f'(x) are developed in memories 8 and 9.

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Data, store	2 [STO] 1	2.00	Store a_0
2	Data, store	5 [CHS] [STO] 2	-5.00	Store a_1
3	Data, store	35 [STO] 3	35.00	Store a_2
4	Data, store	15 [CHS] [STO] 4	-15.00	Store a_3
	Continue for any further coefficients, the example has none			
5	Data, push	1 [ENT] [ENT] [ENT]	1.00	Fill stack with x_i
6	Clear and save	[CLX] [STO] 8 [STO] 9	0.00	Initialize sums
7	Pop	[CLX] [+]	1.00	Pop zero off stack; CLX not needed first pass
8	Data, power	3 [y^x]	1.00	Degree of term as power
9	Recall, multiply, push	[RCL] 1 [X] [ENT] [ENT]	2.00	Multiply by coefficient
10	Recall, add, store	[RCL] 8 [+] [STO] 8	2.00	Add to f(x)
12	Exchange, divide	[Y⇔X] [÷]	2.00	Obtain ax^{n-1}
13	Multiply	3 [X]	6.00	Multiply by degree of term
14	Recall, add, store	[RCL] 9 [STO] 9	6.00	Add to f'(x)
	Repeat steps 7-13 for each term in polynomial, using appropriate memory and power each time. Skip steps 8 and 11 when degree = 1. Go on to step 13 after degree = 1.			
15	Pop, recall, add	[CLX] [+] [RCL] 4 [RCL] 8 [+]	17.00	Add 0 degree term to f(x
16	Recall, divide	[RCL] 9 [÷]	0.55	f(x)/f'(x)
17	Subtract	[−]	0.45	x_i-f(x)/f'(x) = new x
18	Roll	[R↓]	1.00	Move new x to bottom of stack
19	Pop	[CLX] [+] [CLX] [+]	1.00	Move new x up in stack
20	Subtract	[−]	-0.55	Difference between iterations

If difference is small enough, stop (pop stack to get x). If difference is too large, return to step 7.

Quadrature (Simpson's Rule)

Quadrature is the approximation of integrals by numerical methods. Simpson's Rule approximates the area under a curve by summing the areas of parabolas through selected points on the curve. Simpson's Rule requires an even number (2m) of intervals of constant size, h, over the dimension of integration. These 2m intervals define 2m + 1 points on the axis. If $y_i = f(x_i)$ then Simpson's Rule is:

$$\text{Area} \approx (1/3)\,h[(y_0 + y_{2m}) + 4(y_1 + y_3 \ldots y_{2m-1}) + 2(y_2 + y_4 + \ldots + y_{2m-2})]$$

We will illustrate by approximating the integral $\int_2^6 \frac{dx}{x}$ using 4 intervals (h=1).

Thus we have: $\text{Area} \approx (1/3)\,[(1/2 + 1/6) + 4(1/3 + 1/5) + 2(1/4)]$

Stack depth used = 4

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Data, reciprocal	3 DSP 1/x	0.33	1/3 constant for step 9
2	Data, reciprocal	2 DSP 1/x	0.50	y_0
3	Data, reciprocal, add	6 DSP 1/x +	0.67	$y_0 + y_4$
4	Data, reciprocal	3 DSP 1/x	0.33	y_1
5	Data, reciprocal add	5 DSP 1/x +	0.53	$y_1 + y_3$
6	Multiply, add	4 X +	2.80	adds $4(y_1 + y_3)$
7	Data, reciprocal	4 DSP 1/x	0.25	y_2
8	Multiply, add	2 X +	3.30	Adds $2y_2$
9	Multiply	X	1.10	Multiply by 1/3
10	Multiply	1 X	1.10	Multiply by h = AREA

Quadrature may also be used when the equation of the curve is not known, but the values at equal sample points are known. Suppose you are considering buying a piece of property that is bounded on one side by a straight road and on the other side by a wandering stream. You are interested in finding the approximate area of the parcel. You measure along the road at even 20 yard intervals, at each interval you measure the distance to the stream. The following table of measurements results:

x	0	20	40	60	80	100	120
y	0	22	41	53	38	17	0

What is the approximate area (in square yards)?

$$\text{AREA} \approx 20/3[(0 + 0) + 4(22 + 53 + 17) + 2(41 + 38)]$$

Stack depth used = 4

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Data	20 ENT	20.00	Enter h
2	Divide	3 ÷	6.67	h/3
3	Data	22 ENT	22.00	(We skip 0+0)
4	Data, add	53 +	75.00	
5	Data, add, multiply	17 + 4 X	368.00	
6	Data	41 ENT	41.00	
7	Data, add, multiply	38 + 2 X	158.00	
8	Add	+	526.00	
9	Multiply	X	3506.67	Multiply by h/3 to obtain AREA

26 COMPLEX NUMBERS

Introduction

The concept of real numbers represented by a number line is a familiar one. The real number system has been extended by the addition of another number line, called the imaginary number line, which passes through zero and is perpendicular to the real number line. The terms 'real' and 'imaginary' are unfortunate, since 'imaginary' numbers are no less real than 'real' numbers. Imaginary and complex numbers are not something mysterious; they are simple logical extensions to the real number system. The entire plane defined by the two number lines represents what are called complex numbers.

There are several forms which are used to represent complex numbers. The first of these is rectangular form. Since a complex number is a point in the plane, we may represent it by the coordinates on the two axes (real and imaginary). To distinguish the real coordinate from the imaginary one, we affix an indicator to the imaginary one. Mathematicians use i as an indicator, while electrical engineers use j (to distinguish it from the i used for current). Some conventions use the indicator as a prefix and some use it as a suffix. Thus i2, 2i, j2, 2j all represent the same imaginary number. We will adopt the convention of the indicator j used as a suffix (e.g. 2j). In rectangular form, we express the complex number as the sum of its real and imaginary parts (e.g., $2 + 3j$).

Points in a plane may be represented in another form, called polar representation. In this form, the point is represented by a magnitude (distance) and by an angle from the reference axis (positive real). The relationship between rectangular and polar form is the following: $a+bj = r\,\mathrm{COS}\theta + r\,\mathrm{SIN}\theta j = r(\mathrm{COS}\theta + \mathrm{SIN}\theta j)$ where $r = a^2+b^2$. The polar form may be expressed as $r\angle\theta$, meaning a magnitude of r at the angle θ.

Another way of expressing the polar form is called the exponential form. The exponential function e^x is defined as the limit of the series:

$$1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} \cdots$$

The trigonometric functions are defined as:

$$\mathrm{COS}\,\theta = 1 - \frac{\theta^2}{2!} + \frac{\theta^2}{4!} \cdots \qquad \mathrm{SIN}\,\theta = \theta - \frac{\theta^3}{3!} + \frac{\theta^5}{5!} \cdots$$

Remembering that the definition of imaginary numbers states that $1j \cdot 1j = -1$, when we substitute θj for x in the exponential and collect terms we get:

$$e^{\theta j} = \mathrm{COS}\,\theta + \mathrm{SIN}\,\theta j$$

Thus the polar form of a complex number can be expressed as $re^{\theta j}$. The r and θ are exactly the same r and θ in the other expression of polar form and thus the actual numbers used in a calculator solution of a complex number problem will be the same for polar and exponential form. The exponential form will be useful in the problems illustrated below to derive simple forms for various complex number operations.

Polar and rectangular forms are each best suited to particular applications. Rectangular coordinates make addition and subtraction of complex numbers quite easy, while multiplication, division, roots and powers are easier to accomplish in polar form. The functions [→POL] and [INV] [→POL] are used to perform the conversion between coordinate forms. In the following problem solutions the form most appropriate for the particular problem is the one illustrated. However, due to the way the problems are set up and the way the conversions work, if the numbers are desired in an alternate form, they can simply be entered and the appropriate conversion performed before the next program step.

Complex Addition and Subtraction

Complex addition and subtraction are done in rectangular mode. The sum (or difference) of two complex numbers is simply the sum (or difference) of the real parts plus the sum (or difference) of the imaginary parts.

$$(a+bj) + (c+dj) = (a+b) + (c+d)j$$

To do sums and differences of complex numbers on the calculator, we make use of the two coordinate summation feature. The imaginary component is entered into Y, the real component into X and [Σ+] used for adding, [INV] [Σ+] used for subtracting. To obtain the sum, [RCL] Σ is used.

Example:

$(3+4j) + (6-3j) - (4+8j) = (5-7j)$

Stack depth used = 2

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Clear	DSP CLR	0.00	
2	Enter	*4* ENT	4.00	Enter imaginary
3	Enter and add	*3* Σ+	3.00	Enter real; add to sum
4	Enter	*3* CHS ENT	-3.00	Enter imaginary
5	Enter and add	*6* Σ+	6.00	Enter real; add to sum
6	Enter	*8* ENT	8.00	Enter imaginary
7	Enter and subtract	4 INV Σ+	4.00	Enter real; subtract from sum
8	Summation	RCL Σ+	5.00	Get real component
9	Exchange	Y⇔X	-7.00	Get imaginary component

Complex Multiply

The product of two complex numbers is defined as follows:

$$re^{\theta j} \cdot se^{\theta j} = rse^{(\theta+\phi)j}$$

That is, the magnitude of the product of two complex numbers is the product of the magnitudes of the factors, and the angle of the product is the sum of the angle of the factors.

For our example, compute $5e^{.93j} \cdot 7.28e^{.28j}$ (angles in radians).

Stack depth used = 4

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Clear and set mode	DSP CLR DSP RAD	0.00.	Sets mode and clear
2	Enter	*.93* ENT	0.93.	Enter θ
3	Enter	*5* ENT	5.00.	Enter r
4	Enter	*.28* CHS ENT	-0.28.	Enter ϕ
5	Enter and exchange	*7.28* Y⇔X	-0.28.	Move r and s together on stack
6	Roll	R↓	7.28.	θ and ϕ together on bottom of stack
7	Multiply	X	36.40.	Obtain rs
8	Roll	R↓	0.93.	Moves θ and ϕ up in stack
9	Add	+	0.65.	Obtain $\theta + \phi$
	to convert result to rectangular form			
10	Roll	R↓ R↓ R↓	36.40.	Moves rs back to X
11	→Rectangular	DSP INV →POL	28.98.	Real component
12	Exchange	Y⇔X	22.03.	Imaginary component

To multiply a series of complex numbers, we can use the Σ+ function if we use logarithms to obtain the product of the magnitudes.

For our example we will use the same problem as above. Although the example shows only two factors, the solution can be used for any number of factors.

Stack depth used = 2

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Clear and set mode	DSP CLR DSP RAD	0.00.	
2	Enter	*.93* ENT	0.93.	Enter angle
3	Enter and ln	5 ln	1.61.	ln of magnitude
4	Sum	+	1.61.	Adds angles multiplies magnitudes
	steps 2, 3, 4 are repeated for each factor, for the example repeat once using -.28 for the angle and 7.28 for the magnitude.			
5	Recall Sum	RCL Σ+	3.59.	ln of magnitude
6	e^x	INV ln	36.40.	Obtain magnitude
7	Exchange	Y⇔X	0.65.	Obtain angle

Complex Divide

The quotient of two complex numbers in polar form is defined as follows:

$$\frac{re^{\theta j}}{se^{\phi j}} = \frac{r}{s}e^{(\theta-\phi)j}$$

That is, the magnitude of a quotient of two complex numbers is the quotient of the magnitudes, and the angle of the quotient is the difference of the angles. For our example, compute $\frac{6.4e^{.9j}}{1.2e^{.75j}}$ (all angles in radians).

Stack depth used = 4

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Clear and set mode	DSP CLR DSP RAD	0.00.	
2	Enter	*.9* ENT	0.90.	Enter angle
3	Enter	*6.4* ENT	6.40.	Enter magnitude
4	Enter	*.75* ENT	0.75.	Enter angle
5	Enter and exchange	*1.2* ENT Y⇔X	0.75.	Enter magnitude, move it down in stack
6	Roll	R↓	1.20.	Moves magnitudes to top of stack
7	Divide	÷	5.33.	Magnitude of quotient
8	Roll and exchange	R↓ Y⇔X	0.75.	Move angles to top of stack, in proper order
9	Subtract	−	0.15.	Angle of quotient
	To convert to rectangular coordinates			
10	Roll	R↓ R↓ R↓	5.33.	Move magnitude to top of stack
11	Convert	DSP INV →POL	5.27.	Real component
12	Exchange	Y⇔X	0.80	Imaginary component

As with complex multiply, we can do complex division using the summation function. For division we use the [INV] [Σ+] function, again using logarithms to obtain the quotient of the magnitudes.

For the example, we will use the same example as the previous complex divide, but the problem solution can be extended to accomodate any number of divisors. Stack depth used = 2

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Clear and set mode	[DSP] [CLR] [DSP] [RAD]	0.00	
2	Enter	.9 [ENT]	0.90.	Enter angle
3	Enter and ln	6.4 [ln]	1.86.	ln of magnitude
4	Sum	[Σ+]	1.86.	First angle and magnitude are added
5	Enter	.75 [ENT]	0.75.	Enter angle
6	Enter and ln	1.2 [ln]	0.18.	ln of magnitude
7	Minus sum	[INV] [Σ+]	0.18.	difference of angles, quotient of magnitudes magnitudes
	steps 5, 6, 7 are repeated for any other factors in denominator. For this example there are no more factors.			
8	Recall sum	[RCL] [Σ+]	1.67.	ln of magnitude
9	e^x	[INV] [ln]	5.33.	magnitude of quotient
10	Exchange	[Y⇔X]	0.15.	Angle of quotient

Complex Reciprocal

From the definition of complex divide, it is easy to find complex reciprocal.

$$\frac{1}{re^{\theta j}} = \frac{1}{r} e^{-\theta j}$$

For our example, compute the reciprocal of $2.9e^{1.2j}$ (angle in radians).

Stack depth used = 2

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Clear and set mode	[DSP] [CLR] [DSP] [RAD]	0.00	
2	Enter and negate	1.2 [CHS] [ENT]	-1.20	Enter angle and negate
3	Enter and reciprocal	2.9 [DSP] [1/x]	0.34	Reciprocal of magnitude

Coordinates are in proper locations for conversion to rectangular form if desired.

Complex Powers and Roots

From the exponential form for complex numbers, the following defines powers of complex numbers:

$$(re^{\theta j})^n = r^n e^{n\theta j}$$

Since roots are simply fractional powers, this definition will serve for complex roots as well. For our first example, compute $(7.2e^{.7j})^3$ Stack depth used = 4

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Clear and set mode	[DSP] [CLR] [DSP] [RAD]	0.00.	
2	Enter	.7 [ENT]	0.70.	Enter angle
3	Enter	7.2 [ENT]	7.20.	Enter magnitude
4	Enter	3 [ENT]	3.00.	Enter power
5	Roll	[R↓]	3.00.	Moves power to bottom of stack
6	Power	[Y^x]	373.25.	Magnitude
7	Roll	[R↓]	0.70.	Move angle up in stack, power moves also
8	Multiply	[X]	2.10.	Angle

To move the coordinates into position to convert to rectangular coordinates, do three Rolls.

To compute an n^{th} root of a complex number, we use 1/n as a power. For our example, compute the square root (n = 2) of $2.3e^{.7j}$.

Stack depth used = 4

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Clear and set mode	[DSP] [CLR] [DSP] [RAD] [CHS] [ENT]	0.00	
2	Enter	.7 [ENT]	-0.70	Enter angle
3	Enter	2.3 [DSP] [1/x] [ENT]	2.30	Enter magnitude
4	Enter and reciprocal	2	0.50	Obtain power from n
5	Roll	[R↓]	0.50	Moves power to bottom of stack, also in top
6	Power	[Y^x]	1.52	Magnitude
7	Roll	[R↓]	-0.70	Move angle up in stack, power moves up also
8	Multiply	[X]	-0.35	Angle

For rectangular coordinates, do three Rolls and then convert

The program above finds only the principal root. For n^{th} roots, there are n-1 others. Each of the roots has the same magnitude and the angle may be computed by $\theta + 2\pi/k$ where θ is the angle of the principal root (in radians), and k ranges from 1 to n-1.

Complex Trigonometric Functions

To compute the trigonometric functions of complex numbers we express the complex number in rectangular form. In the following definitions, all angles are in radians.

Complex Sine $SIN(a+bj) = SIN\ a \times COSH\ b + (COS\ a \times SINH\ b)\ j$

For our example, compute SIN(3 + 4j) Stack depth used = 3

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Clear and set mode	DSP CLR DSP RAD	0.00.	
2	Enter and store	4 STO 1	4.00.	Enter imaginary component; save for step 7
3	Enter and store	3 STO 2	3.00.	Enter real component; save for step 7
4	Sine	SIN	0.14.	SIN a
5	Exchange and Hyper Cosine	Y⇔X HYP COS	27.31.	COSH b
6	Multiply	X	3.85.	Real component of complex sine
7	Recall	RCL 1 RCL 2	3.00.	Recall b, a
8	Cosine	COS	-0.99.	COS a
9	Exchange and Hyper Sine	Y⇔X HYP SIN	27.29.	SINH b
10	Multiply	X	-27.02.	Imaginary component of complex sine

To convert result to polar form, Exchange and convert

Complex Cosine $COS(a+bj) = (COS\ a\ COSH\ b)-(SIN\ a\ SINH\ b)\ j$

For our example, compute COS(3 + 4j) Stack depth used = 3

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Clear and set mode	DSP CLR DSP RAD	0.00	
2	Enter and Store	4 STO 1	4.00.	Enter b, save for step 7
3	Enter and Store	3 STO 2	3.00.	Enter a, save for step 7
4	Cosine	COS	-0.99.	COS a
5	Exchange and Hyper Cosine	Y⇔X HYP COS	27.31.	COSH b
6	Multiply	X	-27.03.	Real component of complex cosine
7	Recall	RCL 1 RCL 2	2.00.	Recall b, a
8	Sine	SIN	0.14.	SIN a
9	Exchange and Hyper Sine	Y⇔X HYP SIN	27.29.	SINH b
10	Multiply and negate	X CHS	-3.85	Imaginary component of complex cosine

To convert result to polar form, exchange and convert.

Complex Tangent $TAN(a+bj) = \frac{SIN\ 2a}{COS\ 2a + COSH\ 2b} + \frac{SINH\ 2b}{COS\ 2a + COSH\ 2b}\ j$

For our example, compute TAN(3 + 4j) Stack depth used = 3

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Clear and set mode	DSP CLR DSP RAD DSP 4	0.0000.	
2	Enter, multiply and store	4 ENT 2 X STO 2	8.0000.	Enter b, save 2b
3	Enter, multiply and store	3 ENT 2 X STO 2	6.0000.	Enter a, save 2a
4	Cosine	COS	0.9602.	COS 2a
5	Exchange, and Hyper Cosine	Y⇔X HYP COS	1490.4393.	COSH 2b
6	Add and store	+ STO 3	1491.4393.	Save denominator
7	Recall and Sine	RCL 2 SIN	-0.2794.	SIN 2a
8	Exchange and divide	Y⇔X ÷	-0.0002.	Real component of complex tangent
9	Recall and Hyper Sine	RCL 1 HYP SIN	1490.4788	SINH 2b
10	Recall and divide	RCL 3 ÷	0.9994	Imaginary component of complex tangent

To convert result to polar form, exchange and convert.

27 VECTORS

Introduction

The concept of vectors has several interpretations and many useful and varied applications. The more general characterization of a vector is an ordered sequence of values, as $(a_1, a_2, \ldots, a_n)$. An algegraic interpretation of such a sequence is as the coefficients of a linear equation of n variables. A geometric interpretation is that the values are coordinates of a point in n-dimension space. Another characterization of a vector is a quantity that is determined by magnitude and a direction. Examples of such quantities are distance, force, velocity, acceleration. For $n = 2$, both characterizations describe a point in a plane, or a magnitude and direction in a plane. The relationship between rectangular and polar coordinates describes the relationship between these two characterizations of 2-dimension vectors.

In this section, we will describe some basic algebraic operations that are performed on general vectors. Following that, we will illustrate the use of 2-dimensional vectors and solve problems of distance and force. The section on complex numbers details another common and useful application of 2-dimension vectors.

In the illustration of 2-dimension vectors, we will describe three basic types of operations. First, when dealing with vectors that represent physical dimensions, we will illustrate determination of the vector coordinates when the problem definition does not give a specification which is entirely in one vector form or the other. For example, the statement of a problem may give the magnitude and one rectangular coordinate. A second type of operation uses the relationship between polar and rectangular form to find unknown dimensions or angles, when the vector is defined in either polar or rectangular form. A third operation will be illustrated for force vectors. To determine net force acting at a point, we will form the vector sum of all forces acting at that point. To do this we convert the vector to rectangular coordinate form (called resolving the force into X and Y, or horizontal and vertical, components) and add the coordinates to obtain the rectangular form of the net vector. Then conversion to polar form gives the net force as a magnitude and direction.

Vector Addition

The sum of two vectors $(a_1, a_2, a_3, \ldots, a_n)$ and $(b_1, b_2, b_3, \ldots, b_n)$ is:

$(a_1 + b_1, a_2 + b_2, a_3 + b_3, \ldots, a_n + b_n)$ For the summation of two-dimensional vectors, see the program for addition of complex numbers in the complex number section. For more than two dimensions, a different approach is required. For our example, add the two vectors (3.1, 2.0, 5.3) and (.45, 6.2, 7.9).

Two basic approaches are possible. One method of computing the sum is to sum the first coordinates, then sum the second coordinates, etc. This is a straightforward way to compute the sum and is easily extended to the sum of more than one vector and can accomodate vectors of any dimension. Using this method on our example:

Stack depth used = 4*

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Data	*3.1* ENT	3.10	Enter a_1
2	Data and add	*.45* +	3.55	Add b_1, first coordinate of sum
3	Data	*2* ENT	2.00	Enter a_2
4	Data and add	*6.2* +	8.20	Add b_2, second coordinate of sum
5	Data	*5.3* ENT	5.30	Enter a_3
6	Data and add	*7.9* +	13.20	Add b_3, third coordinate of sum

For vectors of greater dimension, the sequence is extended for each additional coordinate. For the sum of more than two vectors, the sum for each coordinate (e.g. step 1,2) is extended to sum the corresponding coordinates for each vector.

* Stack depth used can be reduced to 2 by pressing CLX after each coordinate is determined.

Another approach, applicable only to vectors of dimension nine or less, is to use the memories to store the sum of corresponding coordinates. The sum is then developed by entering and summing data on a vector-by-vector basis, rather than on a coordinate by coordinate basis, as above. For our example:

Stack depth used = 4*

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Data and store	*3.1* STO 1	3.10	Store first coordinate of first vector
2	Data and store	*2* STO 2	2.00	Store second coordinate of first vector
3	Data and store	*5.3* STO 3	5.30	Store third coordinate of first vector
		Continue for each coordinate; the example has three coordinates.		
4	Data and recall	*.45* RCL 1	3.10	Enter first coordinate of next vector, recall previous coordinate sum
5	Add and store	+ STO 1	3.55	Store new sum
6	Data and recall	*6.2* RCL 2	2.00	Enter second coordinate of next vector, recall precious coordinate sum
7	Add and store	+ STO 2	8.20	Store new sum
8	Data and recall	*7.9* RCL 3	13.20	Enter third coordinate of next vector, recall previous coordinate sum

Continue for each coordinate; the example has three coordinates.

Repeat steps 4 through 8 for any additional vectors.

* Stack depth used can be reduced to 2 if CLX is pressed after each store.

Inner (or Dot) Product

For two vectors $(a_1, a_2, a_3, \ldots, a_n)$, and $(b_1, b_2, b_3, \ldots, b_n)$, the Inner Product is a scalar defined as: $a_1b_1 + a_2b_2 + a_3b_3 + \ldots + a_nb_n$

For example, compute the Inner Product of (3.1, 2.0, 5.3) and (.45, 6.2, 7.9).

Stack depth used = 3

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Clear	DSP CLR	0.00	Clear stack, initialize to zeroes
2	Data	*3.1* ENT	3.10	Enter a_1
3	Data and multiply	*.45* X	1.40	Enter b_1 and multiply
4	Add	+	1.40	Add to previous result

Repeat steps 2, 3, 4 for each coordinate. In this example repeat two more times first with a = 2.0, b = 6.2, then with a = 5.3, b = 7.9.

55.67 Inner product

Vector Cross Product

The vector cross product of two vectors (a_1, a_2, a_3) and (b_1, b_2, b_3) is defined as: $(a_2b_3-a_3b_2, a_3b_1-a_1b_3, a_1b_2-a_2b_1)$

For example, compute the cross product of (3.1, 2.0, 5.3) and (.45, 6.2, 7.9).

Stack depth used = 4

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Data, store	*2* STO 2	2.00	Enter a_2 and save
2	Data, store, multiply	*7.9* STO 6 X	15.80	Enter and save b_3; a_2b_3 on stack
3	Data, store	*5.3* STO 3	5.30	Enter a_3 and save
4	Data, store, multiply	*6.2* STO 5	32.86	Enter and save b_2; a_3b_2 on stack
5	Subtract	−	-17.06	First coordinate of cross product
6	Data, store	*.45* STO 4	0.45	Enter b_1 and save
7	Recall, multiply	RCL 3 X	2.39	Recall a_3, a_3b_1 on stack
8	Data, store	*3.1* STO 1	3.1	Enter a_1 and save
9	Recall, multiply	RCL 6 X	24.49	Recall b_3, a_1b_3 on stack
10	Subtract	−	-22.11	Second coordinate of cross product
11	Recall, recall multiply	RCL 1 RCL 5 X	19.22	a_1b_2 on stack
12	Recall, recall multiply	RCL 2 RCL 4 X	0.90	a_2b_1 on stack
13	Subtract	−	18.32	Third coordinate of cross product

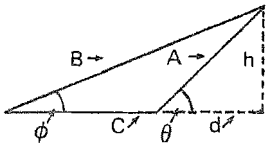
A Simple Boom

Many considerations involving vectors will be illustrated by this simple boom problem. It is called simple not because the problem is simple but because the boom is. The problem will be presented in three parts. The first part will illustrate determination of distance vector coordinates when the problem specification is not completely in either rectangular or polar form. We solve a side-side-side triangle to determine the dimension vectors, and then use conversion to polar form to obtain angles. Part II uses trigonometry to resolve force vectors. Part III uses polar to rectangular conversion to determine force vectors.

PART I

A boom is supported by a guy wire. The length of the boom is A, the length of the guy is B, and the guy is attached a distance C from the base of the boom. What is the height of the end of the boom, and at what angles are the boom and the guy?

Construct the right triangle:



$$h^2 + d^2 = A^2$$

$$h^2 + (C+d)^2 = B^2$$

solve for d: $$d = \frac{B^2 - A^2 - C^2}{2C}$$

then $$h = \sqrt{A^2 - d^2}$$

(C+d, h) are rectangular coordinates that correspond to polar coordinates $B\angle\theta$
Similarly, (d, h) corresponds to $A\angle\theta$
For the example, assume A = 8 B = 15 C = 9

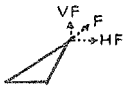
STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Data, square	*15* ENT X	225.00	Square B
2	Data, square	*8* ENT X	64.00	Square A (B^2 still on stack)
3	Store	STO 1	64.00	Save A^2
4	Subtract	−	161.00	Obtain B^2-A^2
5	Data, store	*9* ENT STO 2	9.00	Save C
6	Square	X	81.00	Square C
7	Subtract	−	80.00	Obtain $B^2-A^2-C^2$
8	Recall, multiply	RCL 2 2 X	18.00	Obtain 2C
9	Divide	÷	4.44	Obtain $d = \frac{B^2-A^2-C^2}{2C}$
10	Store	STO 3	4.44	Save d
11	Square	ENT X	19.75	Obtain d^2
12	Recall, subtract	RCL 1 − CHS	44.25	Obtain A^2-d^2 as $-(d^2-A^2)$
13	Square root	DSP √X	6.65	Obtain h (height of boom)
14	Push	ENT ENT	6.65	Push h onto stack for step 18
15	Recall, add	RCL 3 RCL 2 +	13.44	Recall d, C; Obtain C+d
16	Convert to polar	DSP →POL	15.00	Obtain $B\angle\phi$ from (C + d, h)
17	Roll	R↓	26.32	Get ϕ (angle of guy)
18	Roll	R↓	6.65	Get h from stack from step 14)
19	Recall	RCL 3	4.44	Recall d
20	Convert to polar	DSP →POL	15.00	Obtain $A\angle\theta$ from (d, h)
21	Exchange	Y⇔X	56.25	Get θ (angle of boom)

PART II

In the first part of this problem, the angles ϕ and θ were determined. If a weight W hangs from the end of the boom, what is the tension in the guy?

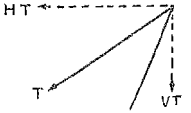
The forces at the end of the boom are compression of the boom, tension in the guy, and the weight.

The compression, F, resolves into force vectors HF and VF.



$$HF = F \cos\theta$$
$$VF = F \sin\theta$$

The tension, T, resolves into force vectors HT and VT.



$$HT = T \cos\theta$$
$$VT = T \sin\theta$$

Since the boom does not move, the forces must be balanced. Thus

$$W + VT = VF \text{ and } HT = HF$$

Solving for F: $$F = \frac{T \cos\phi}{\cos\theta}$$

Then $$W + T \sin\phi = \frac{T \cos\phi}{\cos\theta}$$

$$\sin\theta = T \cos\phi \tan\theta$$

$$T = \frac{W}{\cos\phi \tan\theta - \sin\phi}$$

For the example, assume W = 500

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Enter	*500* ENT	500.00	Enter weight (for step 10)
2	Enter	*26.32* STO 1	26.32	Enter ϕ (from part I)
3	COS	COS	0.90	$\cos\phi$
4	Enter	*56.25* STO 2	56.25	Enter θ (from part I)
5	TAN	TAN	1.50	$\tan\theta$
6	Multiply	X	1.34	$\tan\theta \cos\phi$
7	Recall	RCL 1	26.32	Recall ϕ
8	SIN	SIN	0.44	$\sin\phi$
9	Subtract	−	0.90	$\cos\phi \tan\theta - \sin\theta$
10	Divide	÷	556.75	Obtain T
	To find compression in boom:			
11	Recall and COS	RCL 1 COS	0.90	$\cos\phi$
12	Multiply	X	499.03	$T \cos\phi$
13	Recall	RCL 2 COS	0.56	$\cos\theta$
14	Divide	÷	898.23	Obtain F

PART III

If the guy has a maximum safe tension load of 1000 lbs., what is the maximum weight that may be supported? In the first section of this problem, we used rectangular to polar coordinate conversion to determine lengths and angles. We may use the relationship between polar and rectangular coordinates to resolve force vectors as well. We know the maximum tension is 1000 lbs. at angle ϕ (determined in part I). This represents a polar force vector that can be converted to rectangular form to give HT and VT. We know that HT = HC. Thus we can easily determine C(C=HC/COS θ) and from C we can obtain VC. The maximum safe weight is VC-VT.

Stack depth used = 3

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Enter	*26.32* [ENT]	26.32	Enter ϕ
2	Enter and convert	*1000* [DSP] [INV] [→POL]	896.33	Obtain HT, VT on stack
3	Enter, Store, COS	56.25 [STO] 1 [COS]	0.56	COS θ; HT, VT pushed
4	Divide	[÷]	1613.35	Obtain C
5	Recall	[RCL] 1	56.25	Recall θ
6	Exchange and convert	[Y⇔X] [DSP] [INV] [→POL]	896.33	Obtain HC (=HC), VC
7	Roll	[R↓]	1341.46	Move VC to top of stack, VT in stack from step 3
8	Subtract	[−]	898.07	Obtain VC-VT as -(VT-VC) = max weight

28 ENGINEERING/SCIENTIFIC APPLICATIONS

Skin Diving Depth

We are interested in determining the maximum safe depth that a diver may descend without underwater breathing apparatus. This limit is based on the mechanical effects of pressure on the body – specifically the minimum volume that the lungs may be compressed to by the increased pressure (underwater breathing apparatus compensates for this by providing air at higher pressure to prevent excessive compression). To find the limit of compression, we assume that air behaves as a perfect gas and thus follows the ideal gas laws. We will use Boyle's Law, which states that at a constant temperature, PV = k, where P = pressure, V = volume, and k is a constant. Thus volume is inversely proportional to pressure. To reduce volume to 1/n of original, pressure must be increased n times.

The capacity of human lungs when full is about 12 pints. The minimum capacity is about 3 pints. Thus the safe reduction in volume is about 1/4, which implies that the safe diving depth is limited to about four times pressure increase.

To determine what this depth is in sea water, we note that at the surface, air pressure is about 14.7 lb/in^2 (called one atmosphere or atm). Thus the diving depth is where the pressure is about 4 atm. Since the air at the surface is already 1 atm, we need to determine the depth of water that corresponds to 3 atm of pressure (sea water weighs about 64.2 lb/ft^3).

Stack depth used = 3

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Data	*14.7* [ENT]	14.70	Enter atm in psi
2	Data, square, multiply	*12* [ENT] [X] [X]	2116.80	Convert atm to lb/ft^2
3	Data and divide	*64.2* [÷]	32.97	Enter weight of sea water in lb/ft^3; divide to get atm in feet of water
4	Multiply	*3* [X]	98.92	Depth for 3 atm

Parallel Resistance – D.C. Circuit

The formula for total effective resistance of a parallel circuit of resistors (for D.C. current) is:

$$R_T = \frac{1}{\frac{1}{R_1} + \frac{1}{R_2} + \ldots + \frac{1}{R_n}}$$

Find the effective resistance of a 5 ohm, 10 ohm, and 30 ohm resistor connected in parallel.

Stack depth used = 2

For additional resistors, repeat step 3.

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Data and reciprocal	*5* DSP 1/x	0.20	$1/R_1$
2	Data, reciprocal, add	*10* DSP 1/x +	0.30	Add $1/R_2$
3	Data, reciprocal, add	*30* DSP 1/x +	0.33	Add $1/R_3$
	For additional resistors, repeat step 3			
4	Reciprocal	1/x	3.00	Effective total resistance

Impedance in a Series Circuit – A.C. Current

The A.C. impedance of a series resistance and inductance circuit is given by:

$V = ZI$ Where V is the transform of the voltage
I is the transform of current
Z is the complex impedance

If the voltage function is $v = v_m \cos(wt + \theta)$ then $V = \frac{v_m}{\sqrt{2}} e^{\theta j}$ similarly, for the current function $i = i_m \cos(wt + \phi)$ then $I = \frac{i_m}{\sqrt{2}} e^{\phi j}$
Complex impedance $Z = R + wLj$ where R is resistance and L is inductance, and $w = 2\pi$ frequency.

The resistance of a coil of wire is 1.75 ohms, and the inductance is 5.5 millihenrys. What is the impedance to 60 cycle current? What is the voltage if the current is $i = 5.35 \cos wt$?

Stack depth used = 4

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Data and multiply	*60* ENT 2 × DSP	376.99	Obtain w
		π ×	2.07	Obtain wL; imaginary
2	Data and multiply	*5.5* EE *3* CHS ×	2.07	Obtain wL; imaginary component of Z
3	Data, convert polar	*1.75* DSP →POL	2.71	Enter real component; convert to polar form
4	Data	*5.35* ENT	5.35	Enter i_m
5	Divide	2 DSP √x ÷	3.78	Magnitude of I; reference phase angle of 0
6	Multiply	×	10.26	Magnitude ZI = magnitude V
7	Multiply	2 DSP √x ×	14.52	Obtain v_m
8	Exchange	Y⇔X	49.84	Angle θ of voltage (angle of Z + 0)

Voltage function is $v = 14.52 \cos(wt + 49.84°)$

Decibels

The definition of decibel, db, is given in terms of power ratios:

$$db = 10 \log \frac{P_2}{P_1}$$ Where P_1 = input power, P_2 = output power

When power is expressed as V^2/R, where V = voltage and R = resistance, we get

$$db = 10 \log \frac{V_2^2/R_2}{V_1^2/R_1} = 20 \log \frac{V_2}{V_1} + 10 \log \frac{R_1}{R_2}$$

When $R_1 = R_2$ $db = 20 \log \frac{V_2}{V_1}$

You have an amplifier that has an input impedance of 50,000 ohms and an output impedance of 600 ohms. If an input signal of 4.5 mvolts results in an output of 3.5 volts, what is the gain in db?

Stack depth used = 3

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Data	*3.5* ENT	3.50	Enter V_2
2	Data and divide	*4.5* EE *3* CHS ÷	777.78	Enter V_1; obtain voltage ratio log of voltage
3	Log	DSP log	2.89	log of voltage ratio
4	Multiply	20 ×	57.82	
5	Data	*50* EE *3* ENT	50000.00	Enter R_1
6	Data and divide	*600* ÷	83.33	Enter R_2; obtain Resistance ratio
7	log	DSP log	1.92	log of resistance ratio
8	Multiply	10 ×	19.21	
9	Add	+	77.03	Gain in db

Scaling Factor

You are constructing a piece of digital electronic equipment. The basic internal clock is 18.432 MHz. You require a clock of 76.8 KHz. What is the division factor necessary, and is it an integer (you want the result in business display mode)?

Stack depth used = 2

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Clear and set display	CLX DSP INV SCI	0.	
2	Data	*18.432* EE *6* ENT	18432000.	Mega- $=10^6$; auto conversion to business mode
3	Divide	*76.8* EE *3* ÷	240.	Obtain scaling factor

It is an integer

It is an integer

Straight Line Motion — Constant Acceleration

An automobile is traveling at a speed of 55 miles per hour. It decelerates (negative acceleration) at a rate of 6 ft/sec² when the brakes are applied. What will be the stopping distance?

The applicable equation of linear motion is:

$$v_x^2 = v_0^2 + 2a(x-x_0)$$ where v_0 = initial velocity
v_x = velocity at point x
x_0 = initial position
x = final position

For this problem, $v_x = 0$ (automobile is stopped), $x_0 = 0$, $v_0 = 45$ mph, x is to be determined, and $a = -6.00$. We rewrite the equation as: $\frac{-v_0^2}{2a} = x$

Stack depth used = 3

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Data	*55* [ENT]	55.00	Enter V_0
2	Multiply	5280 [X]	290400.00	Convert miles to feet
3	Square	60 [ENT] [X]	3600.00	Convert hours to seconds
4	Divide	[÷]	80.67	V_0 in feet per second
5	Square, negate	[DSP] [INV] [√x] [CHS]	-6507.11	Obtain $-V_0^2$
6	Data	*6* [CHS] [ENT]	-6.00	Enter a
7	Multiply, divide	2 [X] [÷]	542.26	Stopping distance in feet (not very good brakes)

CHEMISTRY

Stoichiometry

Mass-mass type stoichiometry problems can be computed utilizing the following format:

Using a balanced chemical reaction equation, identify the given substance, called the limiting reagent. Calculate the number of moles of limiting reagent involved. Identify the desired substance and multiply the number of moles of limiting reagent by the stoichiometric ratio which will give you the number of moles of desired substance involved in the reaction. The stoichiometric ratio is the ratio of moles of desired substance per mole of limiting reagent (obtained from the coefficients shown in the balanced equation). The mass (in grams) of the desired substance can then be computed.

Example: If 51.00 grams of ammonia, NH_3, decomposes to form hydrogen gas and nitrogen gas, find the amount (in grams) of nitrogen gas formed.

$$2NH_3 \rightarrow 3H_2 + N_2$$

$$\frac{51.00\text{ g. } NH_3}{1} \times \frac{1\text{ mole } NH_3}{\text{mol. wt. } NH_3 \text{ (in g.)}} \times \left[\frac{1\text{ mole } N_2}{2\text{ moles } NH_3}\right] \times \frac{\text{mol. wt. } N_2}{\text{mole } N_2}$$

stoichiometric ratio

Atomic weight of:
Hydrogen = 1.01
Nitrogen = 14.0

Stack depth used = 3

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Data	*51.0*	51.0	mass of limiting reagent
2	Data	[ENT] *1.01*	1.01	atomic wt. of Hydrogen
3	Multiply	[ENT] *3* [X]	3.03	wt. of one mole of H_3
4	Add	*14.0* [+]	17.03	molecular wt. of NH_3 (g./mole)
5	Reciprocal	[DSP] [1/x]	0.06	moles per gram NH_3
6	Multiply	[X]	2.99	moles of limiting reagent involved in reaction
7	Data	*1*	1	coefficient of N_2
8	Divide	[ENT] *2* [÷]	0.50	stoichiometric ratio
9	Multiply	[X]	1.50	moles of desired substance
10	Data	*14.0*	14.0	atomic wt. of Nitrogen
11	Multiply	[ENT] *2* [X]	28.00	grams N_2 per mole
12	Multiply	[X]	41.93	mass of desired substance involved (in grams)

General Gas Equation

A general gas equation problem can be computed utilizing the following formula and its corollaries: $P \cdot V = n \cdot R \cdot T$

$P \cdot V = n \cdot R \cdot T$

$n = g/\text{mol. wt.}$

$P \cdot V = \frac{g}{\text{mol. wt.}} \cdot R \cdot T$

$\text{mol. wt.} = \frac{g \cdot R \cdot T}{P \cdot V}$

P = pressure (in atmospheres)*
V = volume (in liters)
T = temperature (in degrees Kelvin – see section 8.2.2.1)
g = number of grams in sample
R - General Gas Constant = 0.082
mol. wt. = number of grams per mole

Example: If 0.20 g of a gas occupies a volume of .82 liters at 2.0 atmospheres of pressure at 27°C, what is the molecular weight of the gas?

Stack depth used = 4

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Data	*.20* ENT	0.20	grams of sample
2	Data	.082 ENT	.08	R entered; display rounded
3	Data	*27* ENT	27.00	Degrees Celsius
4	Add	273.16 +	300.16	Obtain T
5	Multiply	X	24.61	Compute R · T
6	Multiply	X	4.92	Compute g · R · T
7	Data	*2.0 ENT*	2.00	Atmospheres of pressure
8	Multiply	*.82* X	1.64	Compute P · V
9	Divide	÷	3.00	Obtain mol. wt. (grams per mole)

* Solutions can be calculated if pressure is given in torrs, or mmHg simply by using 62.4 as the value for R.

29 SPEEDOMETER-ODOMETER CALCULATIONS

Presuming that your automobile speedometer and odometer are not precisely accurate, there are four corrections that are needed:

1. Find true speed from indicated speed.
2. Find indicated speed from a specified true speed (e.g. posted speed limit).
3. Find actual distance traveled from indicated distance.
4. Find indicated distance to travel a specified true distance.

For simplicity, we assume that the speedometer error is the same as the odometer error, and thus we need only one correction factor based on traveling a measured test distance.

First, we determine the correction factor:

S = mileage at start of test section = 3179.1
T = mileage at end of test section = 3183.9
The test section is true 5 miles long.

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Enter	*3179.1* ENT	3179.10	Enter S
2	Enter and Exchange	*3183.9* Y⇔X	3179.10	Enter T, exchange
3	Subtract	−	4.80	T – S
4	Enter and Divide	*5* ÷ STO 1	0.96	Save correction factor

The conversion factor is now in memory 1. To compute true speed from indicated 60 MPH:

5	Clear	CLX	0.00	
6	Enter and	*60* RCL 1 X	57.60	True speed Note: RCL causes automatic push of data entered.

To calculate indicated speed at true 55 MPH:

7	Clear	CLX		
8	Enter and divide	*55* RCL 1 ÷	57.29	Indicated speed

To find actual distance traveled when start at 41291.2 and go to 41351.7:

9	Clear	CLX DSP 1	0.0	Set display
10	Enter	*41291.2* ENT	41291.2	Enter Start
11	Enter and exchange	*41351.7* Y⇔X	41291.2	Enter End
12	Subtract	−	60.5	Indicated distance
13	Recall and divide	RCL 1 ÷	63.0	Actual distance

To find indicated odometer reading after traveling 32.7 miles from 41792.3:

14	Clear	CLX DSP 1	0.0	Set display
15	Enter and multiply	*32.7* RCL 1 X	31.4	Indicated distance
16	Enter and add	41792.3 +	41823.7	Odometer reading at destination

We assumed that the error in the speedometer was the same as in the odometer. This may not be strictly true. However, it is more difficult to make the measurements needed to determine the correction factor for the speedometer. To do so requires measuring elapsed time to travel a known distance at a constant indicated speed.

Assume that 3 min. and 7.2 sec. are required to travel a measured miles at an indicated speed of 62 MPH.

STEP	FUNCTION	KEYSTROKES	DISPLAY	COMMENTS
1	Clear	CLX DSP 2	0.00	Set display
2	Enter	*3* ENT	3.00	Enter distance
3	Enter	*7.2* ENT	7.20	Enter seconds
4	Divide	60 ÷	0.12	Convert to min.
5	Enter and add	*3* +	3.12	minutes
6	Divide	60 ÷	0.05	convert to hrs.
7	Divide	÷	57.69	get actual MPH
8	Divide	*62* ÷	.93	correction factor

APPENDIX A – CORVUS 500 CORVUS FUNCTION SUMMARY

This section presents a tabular summary of all functions available on tl CORVUS 500. Functions are organized in this section by calculator keys stepping through them as if reading a book. For each function, a list of al affected stack registers or memories is included. The effect of each operatio the keystrokes to cause the function to be performed and any undefine operands are also listed. Except as noted, each function leaves a "pus pending" (see section 3).

A single keystroke sequence is illustrated for each operation. Actually, sever sequences may cause the same operation to be executed. These options a available because the keystrokes DSP , INV , and HYP may (if th occur in the sequence) be rearranged.

DSP

The main purpose of this key is to shift to second functions. Each of thes functions will be described in turn. All that remains is control of round-c and display mode control.

Round-off Control

Keystrokes: DSP followed by any single digit

Effect: The display is rounded to the indicated number of decim places

Display Mode Control

See SCI key.

INV

The inverse key is described with the applicable function.

STO

Memory Store

Keystrokes: STO followed by single digit

Effect: X reg → Mem n where n is indicated digit

Stack unchanged

Memory Exchange

Keystrokes: STO RCL or RCL STO followed by single digit

Effect: X reg→ Mem n
Mem n→ X reg

RCL

Memory Recall

Keystrokes: [RCL] followed by any single digit

Effect: Mem n→ X reg
X reg→ Y reg
Y reg→ Z reg
Z reg→ W reg
W reg→ lost

Memory Exchange

Same as for [STO] key.

Recall Summation

Keystrokes: [RCL] [Σ+]

Effect: Mem 9→ reg = Σx
Σy Mem→ Y reg = Σy
X reg→ Z reg
Z reg→ lost
W reg→ lost

HYP

Hyperbolic Sine

Keystrokes: [HYP] [SIN]

Effect: SINH(X reg)→ X reg

Hyperbolic Tangent

Keystrokes: [HYP] [TAN]

Effect: TANH(X reg)→ X reg

Inverse Hyperbolic Cosine

Keystrokes: [INV] [HYP] [COS]

Effect: $COSH^{-1}$(X reg)→ X reg

Rectangular → Hyperbolic Polar

Keystrokes: [HYP] [DSP] [→POL]

Effect: $\sqrt{(X\ reg)^2 - (Y\ reg)^2}$→ X reg
$TANH^{-1}$ X reg →Y reg
X reg > Y reg

Hyperbolic Polar → Rectangular

Keystrokes: [INV] [HYP] [DSP] [→POL]

Effect: (X reg) COSH(Y reg)→ X reg
(X reg) SINH(Y reg)→ Y reg

Hyperbolic Cosine

Keystrokes: [HYP] [COS]

Effect: COSH(X reg)→ X reg

Inverse Hyperbolic Sine

Keystrokes: [INV] [HYP] [SIN]

Effect: $SINH^{-1}$(X reg)→ X reg

Inverse Hyperbolic Tangent

Keystrokes: [INV] [HYP] [TAN]

Effect: $TANH^{-1}$(X reg)→ X reg

$-1 < value\ in\ X\ reg < 1$

CLX

Clear

This operation performs three distinct functions

1. If the display is flashing indicating an invalid operand or out of range result, [CLX] will stop the flashing and "unlock" the calculator.
2. If you are in the middle of a multi-keystroke function entry and have keyed-in a portion of the function entry except (or in addition to) [DSP] , [CLX] will undo the keystrokes.
3. If neither 1 or 2 above, then [CLX] will zero the display, zero the X reg and kill any pending push.

CLR

Clear Stack

Keystrokes: [DSP] [CLR]

Effect: 0→X reg
0→Y reg
0→Z reg
0→W reg

resets summation memories (see Part I, Section 20.2)

Σ+

Summation Plus

Keystrokes: [Σ+]

Effect: Stack unaffected
((mem 7) + 1)→ mem 7
mem 8 + $(X\ reg)^2$→ mem 8
mem 9 + (X reg)→ mem 9
Σy mem + (Y reg)→ Σy mem
no push left pending

Summation Minus

Keystrokes: [INV] [Σ+]

Effect: Stack unaffected
((mem 7)- 1)→ mem 7
mem 8 - $(X\ reg)^2$→ mem 8
mem 9 - (X reg)→ mem 9
Σy mem - (Y reg)→Σy mem
no push left pending

Recall Summation

See [RCL] key

$\bar{x}$,s

Mean and Standard Deviation

Keystrokes: [DSP] [$\bar{x}$,s]

Effect: $\frac{mem\ 9}{mem\ 7}$→ X reg = x

$$\sqrt{\frac{mem\ 8 - \frac{(mem\ 9)^2}{mem\ 7}}{(mem\ 7) - 1}} \rightarrow Y\ reg = \text{standard deviation}$$

X reg→ Z reg
Y reg→ W reg
Z reg→ lost
W reg→ lost

ln

Natural Logarithm

Keystrokes: ln

Effect: ln(X reg)→ X reg

value in X reg > 0

Exponentiation (e^x)

Keystrokes: INV ln

Effect: $e^{(X\ reg)}$ → X reg

log

Common Logarithm

Keystrokes: DSP log

Effect: log(X reg)→ X reg

Power of Ten

Keystrokes: INV DSP log

Effect: $10^{(X\ reg)}$ → X reg

SIN

Sine

Keystrokes: SIN

Effect: SIN(X reg)→ X reg

value in X reg is interpreted as degrees or radians depending on mode

Inverse Sine

Keystrokes: INV SIN

Effect: SIN^{-1}(X reg)→ X reg

-1≤value in X reg before operation≤1

value in X reg after operation is expressed in either degrees or radians depending on mode

Hyperbolic Sine

See HYP key.

Inverse Hyperbolic Sine

See HYP key.

→ POL

Rectangular → Polar

Keystrokes: DSP →POL

Effect: $(X\ reg)^2 + (Y\ reg)^2 \rightarrow X\ reg = r$

$$TAN^{-1}(Y\ reg) \rightarrow \frac{Y\ reg}{X\ reg} = \theta$$

θ is expressed in either degrees or radians depending on mode.

Polar → Rectangular

Keystrokes: INV DSP →POL

Effect: (X reg) COS(Y reg)→ X reg

(X reg) SIN(Y reg)→ Y reg

value in Y reg before operation is interpreted as either degrees or radians depending on mode

Rectangular → Hyperbolic Polar

See HYP key

Hyperbolic Polar → Rectangular

See HYP key

COS

Cosine

Keystrokes: COS

Effect: COS(X reg)→ X reg

value in X reg before operation is interpreted as degrees or radians depending on mode

Hyperbolic Cosine

See HYP key.

Inverse Cosine

Keystrokes: INV COS

Effect: COS^{-1}(X reg)→ X reg

-1≤value in X reg before operations≤1

value in X reg after operation is in degrees or radians depending on moc

Inverse Hyperbolic Cosine

See HYP key

→RAD

→ Radians

Keystrokes: DSP →RAD

Effect: π (X reg)→ X reg

regardless of mode, X reg value assumed to be degrees, converted to radians

→ Degrees

Keystrokes: INV DSP →RAD

Effect: 180 (X reg)→ X reg

regardless of mode,

X reg value assumed to be radians,

TAN

Tangent

Keystrokes: TAN Effect:

TAN(X reg)→ X reg

value in X reg before operation ≠ 0 + 180n° (where n is any integer). Value in X reg before operation is interpreted as degrees or radians depending on mode

Inverse Tangent

Keystrokes: INV TAN

Effect: TAN^{-1}(X reg)→ X reg

value in X reg after operation is expressed in degrees or radians depending on mode

Hyperbolic Tangent

See HYP key

Inverse Hyperbolic Tangent

See HYP key.

RAD

Radian Mode

Keystrokes: DSP RAD

Effect: Stack unaffected

All angles are subsequently expressed and interpreted in radians

Degree Mode

Keystrokes: INV DSP RAD

Effect: Stack unaffected

all angles are subsequently expressed and interpreted in degrees

Y^X

Power

Keystrokes: Y^X

Effect: $(Y\text{ reg})^{(X\text{ reg})} \rightarrow$ X reg
Z reg → Y reg
W reg → Z reg
W reg → W reg

value in Y reg before operation> 0

$\sqrt{x}$

Square Root

Keystrokes: DSP $\sqrt{x}$

Effect: $\sqrt{}$ (X reg)→ X reg

value in X reg before operation> 0

$Y \leftrightarrow X$

Exchange Registers

Keystrokes: $Y \leftrightarrow X$

Effect: Y reg→ X reg
X reg→ Y reg

%

Percentage

Keystrokes: DSP %

Effect: (X reg) (Y reg) →X reg

R↓

Roll Stack

Keystrokes: R↓

Effect: Y reg→ X reg
Z reg→ Y reg
W reg→ Z reg

Δ

Percentage Difference

Keystrokes: DSP Δ%

Effect: 100 (X reg - Y reg) → X reg

value in Y reg before operation≠0

Root

Keystrokes: INV Y^X

Effect: $\sqrt[(X\text{ reg})]{(Y\text{ reg})} \rightarrow$ X reg
Z reg → Y reg
W reg → Z reg
W reg → W reg

Value in Y reg before operation>0

Square

Keystrokes: INV DSP $\sqrt{x}$

Effect: $(X\text{ reg})^2 \rightarrow$ X reg

Gross Profit Margin

Keystrokes: INV DSP %

Effect: 100 (Y reg) →X reg

value in X reg before operation≠100

CHS

Change Sign

This operation performs three distinct functions defined by context:

1. After the first digit of a mantissa is entered, but before any key (except digit or . keys) is pressed, CHS negates the mantissa.
2. After EE is pressed but before any other key (except digit keys or . key, which is ignored) is pressed, CHS negates the exponent part of a number.
3. After any operation is complete, CHS negates the value in the X reg.

X!

Factorial

Keystrokes: DSP X!

Effect: (X reg)!→ X reg

value in X reg before operation must be integer≠0

value in X reg before operation>69; result out of normal range

value in X reg before operation>120; result out of extended range

EE

Enter Exponent

This operation performs two distinct functions based on context:

1. Immediately after keying-in a mantissa, EE will indicate the start of keying-in an exponent (i.e. power of ten).
2. Any other time, EE will cause a 1 to be placed into the X reg as a mantissa and will indicate the start of keying-in an Exponent. If a push is pending, the stack will be pushed.

1/x

Reciprocal

Keystrokes: DSP 1/x

Effect: $\frac{1}{(X\text{ reg})} \rightarrow$ X reg

value in X reg before operation≠ 0

÷

Divide

Keystrokes: ÷

Effect: $\frac{X\text{ reg}}{(Y\text{ reg})} \rightarrow$ X reg
Z reg → Y reg
W reg → Z reg
W reg → W reg

value in X reg before operation≠0

KG→LB

Kilograms to Pounds

Keystrokes: DSP KG→LB

Effect:

2.20462262185 x (X reg)→X reg

Pounds to Kilograms

Keystrokes: INV DSP KG→LB

Effect:

0.45359237 × (X reg)→X reg

LAST X

Last X

The LAST X key is used to access a special purpose memory. RCL LAST X recalls the X reg operand from the last operation performed and places it in the X reg.

π

Pi

Keystrokes: DSP π

Effect: 3.14159265359 → X reg
X reg → Y reg
Y reg → Z reg
Z reg → W reg
W reg → lost

ENT

Enter (or Modified Push)

Keystrokes: ENT

Effect: X reg → X reg
X reg → Y reg
Y reg → Z reg
Z reg → W reg
W reg → lost

SCI

Set Display Format

In conjunction with DSP, SCI is utilized to control the format of the display. DSP SCI causes the display to show results in scientific notation without roundoff and with zeroes suppressed. DSP INV SCI causes the display to show results in business notation without roundoff and with zeroes suppressed. (see also DSP)

C→F

Centigrade → Fahrenheit

Keystrokes: DSP C→F

Effects: $\frac{9}{5}$(X reg) + 32→ X reg

Fahrenheit → Centigrade

Keystrokes: INV DSP C→F

Effect: $\frac{5}{9}$ (X reg - 32)→ X reg

X

Multiply

Keystrokes: X

Effect: (X reg) × (Y reg)→ X reg
Z reg → Y reg
W reg → Z reg
W reg → W reg

LTR→GAL

Liters to Gallons

Keystrokes: DSP LTR→GAL

Effect:

0.264179449175 × (X reg)→X reg

Gallons to Liters

Keystrokes: INV DSP LTR→GAL

Effect:

3.7830579544 × (X reg)→X reg

−

Subtract

Keystrokes: −

Effect: (Y reg) - (X reg)→ X reg
Z reg → Y reg
W reg → Z reg
W reg → W reg

CM→IN

Centimeters to Inches

Keystrokes: DSP CM→IN

Effect:

9.39370078702 × (X reg)→X reg

Inches to Centimeters

Keystrokes: INV DSP CM→IN

Effect:

2.54 × (X reg)→X reg

+

Add

Keystrokes: +

Effect: (X reg) + (Y reg)→ X reg
Z reg → Y reg
W reg → Z reg
W reg → W reg

APPENDIX B – USING THIS BOOK WITH OTHER CALCULATORS

All functional descriptions of calculator features and all solution programs which appear in this book are oriented toward the CORVUS 500. That fact does not necessarily imply that the book is only valuable to CORVUS 500 owners. It does imply that the differences between the CORVUS 500 and your own scientific calculator need to be carefully catalogued. As long as your calculator utilizes RPN, most of the material in this book should be useful.

Some of the main features which may be different include the stack depth and the number of memories. In addition each manufacturer seems to have his own unique set of idiosyncracies. On the CORVUS 500, for example, the bottom of the stack is copied when the stack is popped. Each calculator also seems to have its own mechanism for summation sequences.

Whenever an identified, and seemingly unique, characteristic of the CORVUS 500 is utilized in a program in part II, that feature is mentioned in the problem introduction. Furthermore, each application problem lists the stack depth required. An asterisk by the stack depth indicates that a special CORVUS feature is utilized in the solution program.

Each calculator will have some functions in common with the CORVUS 500 and other functions which are not in common. The specific keystrokes to cause each function to be performed are sure to be different.

The best way to identify the calculator differences is to create a version of Appendix A for your calculator. That table will provide an equivalent keystroke sequence for each function. More importantly, the table will indicate those functions where the calculators differ. In this way, nearly every solution program can be adjusted by substituting new function keystroke sequences where appropriate. The program logic, including data entry and the arrangement of functions to be performed, remains unchanged.

APPENDIX C – SOME USEFUL CONSTANTS AND FORMULAS

ENGLISH UNITS

12 inches = 1 foot
3 feet = 1 yard
5280 feet = 1 statute mile
1 nautical mile = 1.151 statute miles (at 40 degree latitude)
1 degree latitude = 69 statute miles
1 acre = 43560.0 square feet
1 hectare = 2.471054 acres
1 ft^3 = 0.80357 bushels - 7.84 U.S. gallons
1 Imperial gallon - 1.25 U.S. gallons
1 cup = 8 fluid oz.
1 fluid oz. = 1.80469 $in.^3$
1 pint = 2 cups
1 quart = 2 pints
1 gallon = 4 quarts
1 lb. = 32 oz.
1 ton = 2000 lb.
1 grain = 0.002285 oz.

MISCELLANEOUS CONSTANTS

Plank's constant = $(6.62554 \pm 0.00015) \times 10^{-27}$ erg sec.
Avogadro's number = $(6.02257 \pm 0.00009) \times 10^{23}$ $mole^{-1}$
Mass of hydrogen atom = $(1.67339 \pm 0.00031) \times 10^{-24}$ gram
Acceleration of gravity at sea level = 980.621 cm/$sec.^2$ = 32.1725 ft./$sec.^2$
Velocity of sound in air = 331.36 m./sec. = 1087.1 ft./sec.
Velocity of light in a vacuum = 2.997925×10^{10} cm./sec. = 9.83514×1^{-8} ft./sec.

AREAS, SURFACES, AND VOLUMES

Triangles

with base b and altitude h: area = $\frac{hb}{2}$

with sides A,B,C and opposite angles a,b,c

area = 1 AB SIN c

radius of inscribed circle = $\frac{AB\ SIN\ c}{a+b+c}$

radius of circumscribed circle = $\frac{abc}{8\ AB\ SIN\ c}$

Rectangle with sides a,b

area = ab

Paralleogram with parallel sides a, b and included angle θ

area = ab SIN θ

Trapezoid with parallel sides a, b and altitude h

area = ½(a + b)h

Any quadrilateral with diagonals a,b and angle θ between them

area = ½ ab SIN θ

Regular polygon with n sides of length L

area = $\frac{1}{4}$ n L^2 COT $\frac{180°}{n}$

radius of inscribed circle = $\frac{L}{2}$ COT $\frac{180°}{n}$

radius of circumscribed circle = $\frac{L}{2}$ CSC $\frac{180°}{n}$

Circle with radius r

circumference = $2\pi r$

area = πr^2

length of arc subtended by angle $\theta = \frac{\pi r \theta}{180°}$

length of chord subtended by angle θ = 2r SIN ½θ

area of sector subtended by angle θ = ½sr , where s is arc length

Ellipse with semi-axes a,b

circumference = approx $2\pi \sqrt{\frac{a^2 + b^2}{2}}$

area = πab

Pyramid (right)

volume = ½ area-of-base × altitude

Regular polyhedra with edge length L, and n is the number of surfaces

n	Surface	Volume
4	1.73205 L^2	0.11785 L^3
6 cube	6.00000 L^2	1.00000 L^3
8	3.46410 L^2	0.47140 L^3
12	20.6457 L^2	7.66312 L^3

Sphere with radius r

surface = $4\pi r^2$

volume = $\frac{4}{3}\pi r^3$

Cylinder (right) with radius of base r and altitude h

curved surface = $2\pi rh$

volume = $\pi r^2 h$

Cone (right) with radius of base r and altitude h

curved surface = $\pi r \sqrt{r^2 + h^2}$

volume = $\pi/3\, r^2 h$

TRIGONOMETRIC RELATIONS

For any triangle with sides A,B,C and opposite angle a,b,c:

$$\frac{A}{\text{SIN a}} = \frac{B}{\text{SIN b}} = \frac{C}{\text{SIN c}}$$

SIN 2X = 2 SIN X COS X

COS 2X + COS^2X-SIN^2X = 2 COS^2X - 1 = 1 - 2 SIN^2X

$$\text{TAN } 2X = \sqrt{\frac{2\,\text{TAN X}}{1 - \text{TAN}^2\ \text{X}}}$$

$$\text{SIN } ½X = \pm\sqrt{\frac{1 - \text{COS X}}{2}}$$

$$\text{COS } ½X = \pm\sqrt{\frac{1 + \text{COS X}}{2}}$$

$$\text{TAN } ½\ \text{X} = \pm\sqrt{\frac{1 - \text{COS X}}{1 + \text{COS X}}} = \frac{1 - \text{COS X}}{\text{SIN X}} = \frac{\text{SIN X}}{1 + \text{COS X}}$$

SIN X SIN Y = 2 SIN ½(X Y) COS½(X Y)

COS X + COS Y = 2 COS ½(X+Y) COS ½ (X−Y)

COS X − COS Y = −2SIN ½(X+Y) +COS½(X−Y)

SIN X COS Y = ½ (SIN(X+Y) + SIN (X−Y))

COS X SIN Y = ½(SIN (X+Y) − SIN (X−Y))

COS X COS Y = ½(COS (X+Y) + COS (X−Y))

SIN X SIN Y = ½(COS (X−Y) − COS (X+Y))

INDEX

NOTES

NOTES

NOTES

NOTES

PUBLISHED BY
tk enterprises
BETTER BUSINESS PRODUCTS