CALCULUS

WITH THE HEWLETT-PACKARD SYMBOL-MANIPULATING CALCULATORS

HP-28S AND HP-48SX

LYNN E. GARNER

Calculus
with the Hewlett-Packard
Symbol-Manipulating
Calculators

HP-28S and HP-48SX

Lynn E. Garner

DELLEN PUBLISHING COMPANY, San Francisco
a division of
MACMILLAN PUBLISHING COMPANY, New York

COLLIER MACMILLAN CANADA, Toronto

MAXWELL MACMILLAN INTERNATIONAL
New York Oxford Singapore Sydney

Macmillan Publishing Company
866 Third Avenue, New York, New York 10022

Collier Macmillan Canada, Inc.

Printing: 1 2 3 4 5 6 7 Year: 0 1 2 3 4 5 6

ISBN 0-02-340599-6

Preface

The symbol-manipulating calculators, the HP-28S and HP-48SX, have broken new ground by placing the power of graphics and symbol-manipulation, as well as computation, into a pocket-sized machine. This power is increasingly within the reach of students of the mathematical, engineering, and physical sciences, and is becoming ever more attractive to those in other disciplines. The calculators are so powerful that they are changing the way courses are taught, enabling students and instructors to condense many tedious and time-consuming techniques into a few keystrokes. Insights are being gained into the nature of mathematical and physical processes that are not possible without the computing power that these machines make available.

It is not the purpose of this monograph to explain all the features of the HP symbol-manipulating calculators. Rather, its aim is to give the student a brief tutorial in the basic mathematical uses of the HP-28S and HP-48SX, and then to explore the many ways in which these machines can be used to enhance a calculus course. In the process, the student will become familiar with most of the mathematical features of the machines.

Neither is it the purpose of this monograph to teach calculus; that is left to the calculus course and the textbook. We assume that the student knows how to do the calculus "by hand," and is looking for a way to speed up the process with the calculator. The standard topics of the three-semester "engineering and science" calculus course are dealt with here, organized in more or less the standard order. Ways to use the calculators in each of the topics are presented. Rather than include exercises here, the student is encouraged to apply the calculator to the exercises given in the calculus book.

Therefore, this monograph is a collection of tutorial helps, tips, tricks, explanations, and programs that will aid the calculus student to "get the most" from his investment in the state-of-the-art technology represented by the HP-28S and HP-48SX.

While this tutorial can be used with any study of the calculus, the topics treated here are cross-referenced with the author's calculus text, *Calculus and Analytic Geometry*, also published by Dellen/Macmillan. The section in which a topic is found is given inside a double box, such as that below, near the heading for that topic.

$\boxed{\boxed{1.1}}$

Thanks is due to Clain Anderson and the development team at Hewlett-Packard for their technical help and cooperation, and to Don Dellen and the staff at Dellen/Macmillan for the publication of this book. Thanks also goes to Professor Thomas W. Tucker of Colgate University, who first introduced me to the finer points of programming the HP-28S. And finally, thanks to numerous students and instructors who have been a patient audience and who have shared ideas and discoveries.

Lynn E. Garner
Brigham Young University

Table of Contents

Chapter 1. Basic Use Tutorial

The Keyboard

Take some time to study the keyboard of your machine. Note the locations of the various types of keys: the number keys, the letter keys, and the operation keys. Note other symbols that occur as labels; you will need them eventually. Many of the keys are menu keys, but you can't tell by looking in all cases; you will learn them as we go along.

Of particular importance are the shift keys. On the HP-28S there are two keys that behave as shift keys. The red key is called the *shift key*, and is used to get the label in red associated with each key. For example, to execute $\boxed{\text{OFF}}$, press the shift key and then $\boxed{\text{ON}}$. The other key that behaves like a shift key is marked $\boxed{\text{LC}}$, and is used to get lower case letters. It is almost a shift lock in its action; see pp. 26, 28 of the *HP-28S Owner's Manual.*

On the HP-48SX there are three keys that behave as shift keys. The blue key is called the *right shift key*, and is used to get the blue labels; the orange key is called the *left-shift key*, and is used to get the orange labels. The $\boxed{\alpha}$ key is used to get the letters. The right- and left-shift keys are used with the $\boxed{\alpha}$ key to get other symbols that are not printed on the machine; see pp. 52ff of the *HP-48SX Owner's Manual.*

The HP-48SX also has a User keyboard, activated by the $\boxed{\text{USR}}$ command, that is completely blank; you can assign to the keys anything you want, as described starting on p. 216 of the *HP-48SX Owner's Manual.*

The Stack

The HP-28S and HP-48SX are *stack-based* machines, meaning that the basic method of handling data is by means of a *stack.* Each new item that is entered into the machine becomes the bottom object on the stack, and all other objects already on the stack are pushed up to the next higher positions.

Commands given to the machine are applied to objects already on the stack. For example, the command +, to add, is interpreted by the machine to mean that the bottom two objects on the stack are to be taken from the stack and added, and then their sum is to become the new bottom object. Since two objects are taken off the stack, and only one is placed back on, all other items on the stack will drop down one position.

Entry

To enter an item into the machine, simply type it in and signal the end by pressing the $\boxed{\text{ENTER}}$ key. The item is placed on the bottom of the stack.

If you make a mistake while typing an item, the back arrow key $\boxed{\Leftarrow}$ will erase one character at a time. To erase the entire item you are typing, before you press the $\boxed{\text{ENTER}}$ key, press the $\boxed{\text{ATTN}}$ key (the $\boxed{\text{ON}}$ key).

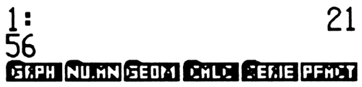
To remove the bottom object from the stack, use $\boxed{\text{DROP}}$ (on the HP-48SX, the $\boxed{\Leftarrow}$ key performs the same function if there is no command line). $\boxed{\text{DROP}}$ may be used repeatedly to remove several objects from the stack, one after another. To remove all objects from the stack, press $\boxed{\text{CLEAR}}$ or $\boxed{\text{CLR}}$.

Operations

As was mentioned above, each command given to the machine applies to the objects already on the stack. The buttons $\boxed{+}$, $\boxed{-}$, $\boxed{\times}$, and $\boxed{\div}$ command the machine to take the bottom two objects from the stack, add, subtract, multiply, or divide, respectively, and put the result back on the stack. Thus to perform an operation on two numbers, you must first place the two numbers on the stack. Actually, each of the arithmetic commands will also enter the second number for you, making it unnecessary to press $\boxed{\text{ENTER}}$ after the second number.

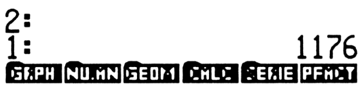
For example, to multiply 21 by 56, type

2 1 $\boxed{\text{ENTER}}$ 5 6



```
1:                21
56
```

$\boxed{\times}$.



```
2:
1:              1176
```

This order of doing things is what is called "reverse Polish" logic.

Other commands work the same way, though they may not operate on exactly two stack objects. The exponentiation command $\boxed{\wedge}$ or $\boxed{y^x}$ takes two numbers from the stack, but the $\boxed{1/x}$, $\boxed{\sqrt{x}}$, and $\boxed{x^2}$ commands operate on only the bottom number, the one in level 1. Still other commands that we will meet take many objects from the stack, and some do not use stack objects. For example, $\boxed{\text{CLEAR}}$ uses as many stack objects as there are, and $\boxed{\text{OFF}}$ ignores the stack.

Stack Manipulation

$\boxed{\text{DROP}}$ and $\boxed{\text{CLEAR}}$ are examples of stack manipulation commands; that is, they only rearrange the objects on the stack. Another stack manipulation command on the keyboard is $\boxed{\text{SWAP}}$, which, as you might expect, interchanges the bottom two objects on the stack.

Another useful command is to press $\boxed{\text{ENTER}}$ to duplicate the object on the bottom of the stack. It is equivalent to the $\boxed{\text{DUP}}$ command found on the STACK or PRG STK menu; DUP is used in a program to duplicate the bottom object.

Other stack manipulation commands are found in the STACK menu, described in the *HP-28S Reference Manual*, beginning on p. 239, or on the PRG STK menu, described in the *HP-48SX Owner's Manual* on p. 78.

Data Types

The operations and manipulation commands given above apply not only to numbers on the stack, but other objects as well. The various types of objects that can be handled on the HP-28S are indicated on the upper left side of the open calculator. The HP-48SX also has these data types, and several more.

To give you an idea of how different objects are treated by the same command, we will consider the $\boxed{+}$ command, addition. If real numbers, complex numbers, or binary integers occupy both the bottom levels of the stack, $\boxed{+}$ has the anticipated effect: it replaces them with their sum. The same is true of vectors and matrices, provided that the arrays have the same sizes; an array sum is a new array whose entries are the sums of the entries in the old arrays. For strings and lists, $\boxed{+}$ means concatenate; for example, if "AB" is added to "CDE", the result is "ABCDE". For names and algebraic expressions, $\boxed{+}$ yields the indicated sum, which is presented as an algebraic expression. On the HP-48SX, if $\boxed{+}$ is applied to two graphics objects, the objects are superimposed. $\boxed{+}$ will not operate on programs.

Results of operations on combinations of object types is described in the *HP-28S Reference Manual*, beginning on p. 53. Types of objects are described briefly on p. 212. If there is a menu with the same name as an object type, more information about that type can be found in the *HP-28S Reference Manual* under the description of that menu. For example, more information about real objects can be found under the REAL menu description, beginning on p. 213.

In the HP-48SX, the data types are described starting on p. 80 of the *HP-48SX Owner's Manual.* More information on operations is given starting on p. 133.

Menus

There are 24 menus of about five types on the HP-28S. Most menus are storage places for commands of various kinds, usually related to a single data type or calculator function. For example, the REAL menu contains commands that apply to real numbers, and the ARRAY menu is meant for dealing with arrays (vectors and matrices). In every case, the commands in a given menu are described in the *HP-28S Reference Manual* under the name of the menu.

On the HP-48SX, there are a total of 60 menus, counting submenus, listed on pp. 697-8 of the *HP-48SX Owner's Manual,* and discussed briefly on pp. 55ff. Menus are not organized alike on the two machines, though both organizations are logical.

Most menus have more than six commands, the maximum number that will be visible at one time. To get the next page of commands, press the $\boxed{\text{NEXT}}$ or $\boxed{\text{NXT}}$ key.

One feature of the HP-28S will undoubtedly entrap the beginner at least once. The $\boxed{\text{MENUS}}$ command (shift $\boxed{\alpha}$) performs a permanent shift on the upper three rows of keys on the left half of the calculator. In that mode, pressing the $\boxed{\text{A}}$ key calls up the ARRAY menu; to get the letter A, you must press *shift* A. Giving the MENUS command again returns the keys to normal operation.

Soft Keys

The top row of keys under the calculator's display window are called *soft keys* and are used to select commands from a menu. When a menu is showing, each soft key has the meaning of the command in the panel of the menu above that key. When a menu changes, the meaning of the soft key also changes.

On the HP-28S, the soft keys become cursor keys when no menu is showing. Cursor keys are used in editing or interacting with a plot. The cursor enable key ❖, next to the shift key, hides whatever menu is showing; pressing the cursor enable key again brings the menu back.

On the HP-48SX, the cursor keys are always active (unless shifted), and a menu is usually showing. The full use of the cursor keys is summarized on pp. 816-821 of the *HP-48SX Owner's Manual.*

Modes

The MODE menu allows you to select the notation in which numbers are presented, whether you want angles in radians or degrees, and so forth. Its commands are discussed in the *HP-28S Reference Manual*, beginning on p. 145, and in the HP-48SX Owner's Manual, starting on p. 220.

It is probably good to start with the standard notation, STD, radian mode, RAD, and CMD, UNDO, LAST, and ML all enabled on the HP-28S. On the HP-48SX, STD, SYM, STK, ARG, CMD, ML, and RAD are recommended.

The USER and VAR Menus

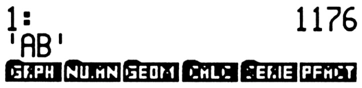
The USER menu on the HP-28S, called the VAR menu on the HP-48SX, is the storage place for containers that you create. Whenever an object is stored in a container other than the stack, it appears on the USER or VAR menu.

Storing

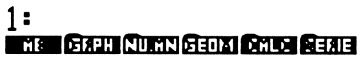
To store an object, put the object on the bottom of the stack. Select a name for the object, and type in the name, beginning with the single quote mark, '. Then press the STO button, and the object is stored in a container with the name you typed.

For example, to store the object on the bottom of the stack in a container named AB, just type

' A B


```
1:                 1176
'AB'
```

and press STO.


```
1:
```

The name AB will appear in the USER or VAR menu.

Recalling

There are several ways to recall the contents of a container. If the container does not contain a program, simply pressing the soft key under the name of the container will put the contents of the container on the stack. If the container does contain a program, however, pressing the soft key will cause the execution of the program. Therefore, a program must be recalled using $\boxed{\text{RCL}}$. The easiest way is to press the quote mark, $\boxed{'}$, and then press the soft key corresponding to the container, thus putting the name of the container on the stack. Then press $\boxed{\text{RCL}}$. The contents of the container is placed on the stack. Using $\boxed{\text{RCL}}$ also works with containers of objects other than programs. On the HP-48SX, a shortcut for recalling the contents of a container is discussed on p. 110 of the *HP-48SX Owner's Manual.*

Purging

To remove a container (and its contents) from the USER or VAR menu, type the name of the container, beginning with $\boxed{'}$, and then press $\boxed{\text{PURGE}}$. To remove several containers at once, make a list of their names, separated by spaces, and then use $\boxed{\text{PURGE}}$. The easiest way to create such a list on the HP-28S is to press $\boxed{\{}$ to start a list, and then press $\boxed{\alpha}$ to change the entry mode so that soft key names are entered with spaces separating them, and then press the soft keys corresponding to the containers that are to be purged. On the HP-48SX, pressing $\boxed{\{\ \}}$ to create a list automatically sets the appropriate entry mode. When the containers to be deleted have all been named, press $\boxed{\text{ENTER}}$ and then $\boxed{\text{PURGE}}$.

Renaming

To change the name of a container, recall the contents and store it under the desired name, and then purge the old container. There is no provision for renaming in any other way.

Editing

There are several ways to change the contents of a container. The first, which is a complete change-over, is simply to store a new object in the old container.

A second way to change the contents of a container is to "visit" the container and edit its contents. First type the name of the container, starting with ', and then press $\boxed{\text{VISIT}}$. The contents of the container is placed in editing mode, and the cursor keys are activated. The cursor can be moved by pressing the arrow keys; the $\boxed{\text{DEL}}$ key deletes the symbol the cursor covers. If you type with the block cursor, previous text is written over; pressing $\boxed{\text{INS}}$ changes the cursor shape to an arrow, after which new text is inserted between previous text. Additional editing mode features are described in the *HP-28S Owner's Manual*, beginning on p. 166, and in the *HP-48SX Owner's Manual*, starting on p. 111.

At the end of editing, pressing $\boxed{\text{ON}}$ discards the edited object; the original contents of the container are unaltered. Pressing $\boxed{\text{ENTER}}$ saves the edited object in the container.

To edit the object in level 1 on the stack, press $\boxed{\text{EDIT}}$. To edit the object in level *n* on the stack, type *n* and then press $\boxed{\text{VISIT}}$. Pressing $\boxed{\text{ENTER}}$ saves the edited object, and pressing $\boxed{\text{ON}}$ discards it, leaving the original unchanged.

Additional editing environments for equations and matrices on the HP-48SX are described in the *HP-48SX Owner's Manual* on pp. 241 and 350.

Ordering

Whenever a new container is created, it appears in the first position on the USER or VAR menu. Therefore, after several containers have been created, you usually discover that they are in the wrong order. Fortunately, they can be reordered, using the $\boxed{\text{ORDER}}$ command on the MEMORY menu.

To use the $\boxed{\text{ORDER}}$ command, first create a list of the menu items in the order in which you want them to appear on the menu. This applies to the entire directory, not just the first six items. Make the list as described in the "Purging" section above. Once the list is created, call up the MEMORY menu and press $\boxed{\text{ORDER}}$; when you return to the USER or VAR menu, the items will be in the order you specified.

If the list you make does not name all the items in the menu, the unnamed items retain their relative order, and are grouped together after the named items in the reordered menu.

Subdirectories

It is soon apparent to the casual observer that the USER or VAR menu can become quite unwieldy. For that reason, it is convenient to create subdirectories, into which related containers can be organized.

To create a subdirectory, type the name you have chosen for it, and then press $\boxed{\text{CRDIR}}$ (for "create directory") on the MEMORY menu. When you return to the USER or VAR menu, you will find the name you typed (or as many letters of it as will show) as first item on the menu. Enter the subdirectory by pressing the soft key below it. You will see a blank menu, waiting for containers to be created. Anything you store at this point will be stored in the subdirectory. You are now down one level from the top in the USER or VAR menu. The top level is called the HOME level; you can return to it by typing the word HOME and entering it, or by pressing the command $\boxed{\text{HOME}}$ (on the MEMORY menu in the HP-28S).

You can also create subdirectories inside subdirectories. Storing takes place in the current level of the USER or VAR menu. You can recall the contents of any container in the current subdirectory, or from any container in any subdirectory "up the line", all the way to the HOME level. Programs stored in the current subdirectory or in any "parent" subdirectory, all the way up to the HOME level, can be "called" or used at any time.

Navigation

With several subdirectories, some of them nested within others, navigation around the USER or VAR menu can become a problem. To find out where you are in the HP-28S USER menu, the [PATH] command from the MEMORY menu is useful. That command returns a list of the names of the subdirectories, from the one you are in (the current level) all the way back up the line to the HOME level. On the HP-48SX, the current path is shown at the top of the display whenever the stack is visible.

If you enter any name from the current path, *without* the single quote mark this time, it is interpreted as a command to go to that subdirectory.

On the HP-48SX, the command [UP] moves you up one level in the current path, to the parent of the current directory. In a program, this command is known as UPDIR.

For the HP-28S, here is a little program that does the same thing. Type in the following program, being careful to observe the spaces:

```
« PATH DUP SIZE 1 - GET EVAL
```

Then press [ENTER], and the program is placed on the stack; the machine supplies the closing program marker, ». Now store the program under the name 'UP' in the HOME level of the USER menu. That is, type ' U P and then press [STO].

Now, whenever you are in a subdirectory, issuing the command UP will take you up one level, to the "parent" directory. If you try to use the program when you are at the HOME level, however, you will get an error message. The program will also have trouble doing what you ask if you name a subdirectory with the same name as its parent directory and then try to leave the lower directory by using UP. A good rule to follow is never to use the same name for two different things.

Here is an idea that provides a very convenient means of issuing the UP command. Whenever you enter a newly-created subdirectory, store the program « UP » in it under the name 'QUIT'. Then simply by pressing the soft key [QUIT], the command UP is issued, and you are taken up one level.

Some Utility Programs

Here are some utility programs that change the form of a number in the machine. They can be used in a variety of contexts to present results in convenient form. If they are in the HOME level of the USER or VAR menu, they will always be available to use.

A remark or two is in order about the entering of programs. One may type in the symbols that make up the program as it is written, or one may select the commands from the various menus. In every case, it is necessary to observe the spacing shown and to distinguish between the numeral 0 and the letter capital O. Spacing is crucial for commands such as →LIST, which involve the arrow and the letters without space between them. If you type the symbols one at a time, there will usually be a space between the arrow and the "L"; then you must go back in edit mode and remove the space. It is better to go to the LIST or PRG OBJ menu and press the key [→LIST] to enter this command into the program. The same is true for such commands as C→R (found on the COMPLEX or PRG OBJ menu) and STO+ (found on the STORE or *right-shifted* MEMORY menu).

Checksum

Another device that aids in correctly typing a program is the checksum, described on p. 101 of the *HP-48SX Owner's Manual.* This is a binary integer uniquely associated to the contents of the program. To determine whether you have typed in the program correctly, after it is stored, put the name of the program on the stack and press BYTES on the MEMORY menu. The check sum and the size of the program are returned; if the checksum is not the same as that given, then there is a typing error in the program.

On the HP-28S there is no checksum built in; here is a program that does the same thing. Store it under the name CHK on the USER menu.

```
« RCLF STD HEX 64 STWS SWAP RCL →STR 16 STWS DUP #0h 1
ROT SIZE FOR j OVER j j SUB NUM R→B XOR RL NEXT →STR 3
OVER SIZE 1 - SUB ROT STOF SWAP DROP »
```

To check this program, enter it and store it. Then type ' C H K ENTER CHK. The message "6A93" should appear. The number "6A93" is the 28S checksum for the program CHK.

Rounding: Number of Decimal Places

The next utility program, N.DEC, tells the machine how many decimal places to take seriously. It can be used to eliminate round-off error in many cases, or to round numbers to a specified accuracy without having to change the mode of the machine or go to a distant menu.

The HP-48SX uses the command RND on the MTH PARTS menu to accomplish this; it is convenient to store the program « RND » in the container N.DEC, or to assign the program to a key on the User keyboard (see p. 217 of the *HP-48SX Owner's Manual*).

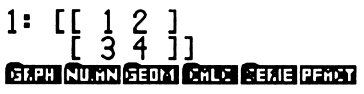
For the HP-28S, here is an equivalent program, necessary because the RND command works differently on the two machines.

```
« RCLF → F « FIX RND F STOF » »
                                        28S checksum: "A636"
```

ENTER this program and store it in N.DEC by typing ' N . D E C STO.

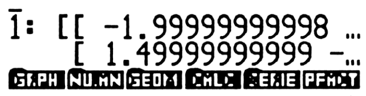
As an example of the use of N.DEC, put your calculator in STD mode, and suppose that you want to invert the matrix $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$.

On the HP-28S, enter the matrix by typing [[1 , 2 [3 , 4 ENTER; on the HP-48SX, enter the matrix by typing MATRIX 1 SPC 2 ENTER ▼ 3 SPC 4 ENTER ENTER.



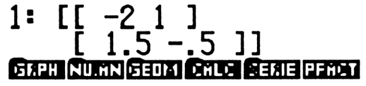
```
1: [[ 1 2 ]
    [ 3 4 ]]
```

Invert it by pressing $\boxed{1/x}$. The result involves numbers with 11 decimal places showing.



```
1: [[ -1.99999999998 …
   [ 1.49999999999 -…
```

Tell the machine to take only 9 decimal places seriously by typing 9 $\boxed{\text{N.DEC}}$ on the HP-28S or 9 $\boxed{\text{RND}}$ on the HP-48SX. Then the result is probably what the machine meant all the time.



```
1: [[ -2 1 ]
   [ 1.5 -.5 ]]
```

Rounding: Number of Significant Digits

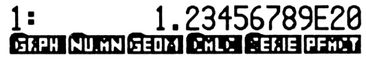
The program N.DEC above will not work on numbers that must be presented in scientific notation because they are too large or too small to be displayed otherwise. Also, it will not give the expected result if your calculator is in SCI or ENG mode. This program, called SIG.D, is a modification of N.DEC that enables you to round the number in level 1 to a specified number of significant digits, thus changing the precision without changing the display mode. For the HP-28S, the program is

```
« RCLF → F « 1 - SCI RND F STOF » »
                                28S checksum: "15BB"
```

For the HP-48SX, the program is

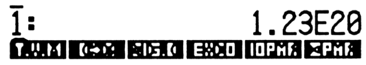
```
« → N « XPON LASTARG MANT N 1 - RND SWAP ALOG * » »
                                48SX checksum: # 28637d
```

For example, if the number 1.23456789E20 is in level 1,



```
1:          1.23456789E20
```

the command 3 $\boxed{\text{SIG.D}}$ rounds it to 3 significant digits.



```
1:                1.23E20
```

Decimals to Fractions

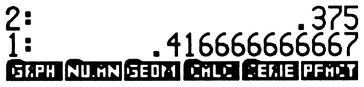
The next program, D→Q, for the HP-28S allows us to rewrite the decimal approximation to a rational number as a fraction. It partially duplicates the command $\boxed{\to\text{Q}}$ found on the HP-48SX. It is not foolproof, but will reliably return fractions with denominators of up to about 4 digits. Here is the program:

```
« DUP DUP IP IF == THEN ELSE → R « R 1 → A B « DO B A
B MOD 'B' STO 'A' STO UNTIL B .00000001 < END A » INV 0
N.DEC DUP R * 0 N.DEC → D N « 'P/Q' 1 N EXSUB 3 D EXSUB
» » END »
                                28S checksum: "75B8"
```

Then press $\boxed{\text{ENTER}}$, and store it by typing ' D → Q $\boxed{\text{STO}}$.

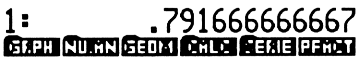
As an example of the use of D→Q, suppose you want to add the fractions $\frac{3}{8}$ and $\frac{5}{12}$. Type the following:

3 $\boxed{\text{ENTER}}$ 8 ÷ 5 $\boxed{\text{ENTER}}$ 12 ÷



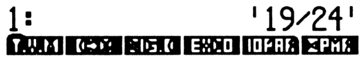
```
2:            .375
1:   .416666666667
```

+



```
1:   .791666666667
```

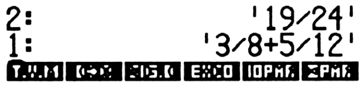
$\boxed{\text{D→Q}}$.



```
1:         '19/24'
```

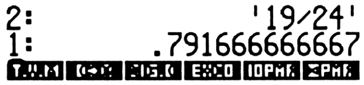
Or you may do it this way:

' 3 ÷ 8 + 5 ÷ 12 $\boxed{\text{ENTER}}$



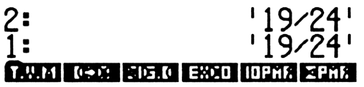
```
2:         '19/24'
1:     '3/8+5/12'
```

$\boxed{\text{EVAL}}$



```
2:         '19/24'
1:   .791666666667
```

$\boxed{\text{D→Q}}$.



```
2:         '19/24'
1:         '19/24'
```

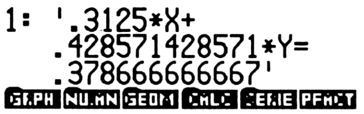
You may do the same thing on the HP-48SX by using $\boxed{\text{→Q}}$ in place of $\boxed{\text{D→Q}}$

The next program is the full equivalent for the HP-28S of the $\boxed{\text{→Q}}$ command on the HP-48SX. It is also called →Q. It takes a real number or an algebraic expression from the stack and replaces the decimal numbers in it with fractions. It is convenient to use when an operation results in an expression with rational coefficients. Here is the program:

« DUP TYPE → E T « IF T 0 == THEN E D→Q ELSE IF T 9 == THEN E SIZE → N « N 1 FOR K E K EXGET DUP TYPE 0 IF == THEN D→Q E SWAP K SWAP EXSUB 'E' STO ELSE DROP END -1 STEP E » ELSE E END END » »

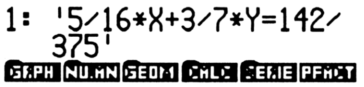
28S checksum: "8599"

Store the program by typing ' → Q $\boxed{\text{STO}}$. For example, if the expression '.3125*X+.428571428571*Y=.378666666667' is on the stack,



```
1: '.3125*X+
   .428571428571*Y=
   .378666666667'
```

then pressing $\boxed{\text{→Q}}$ produces rational coefficients.

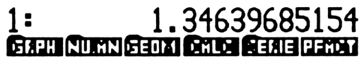


```
1: '5/16*X+3/7*Y=142/
   375'
```

Another program for the HP-28S takes a decimal number from the stack and expresses it as a fraction times π. It will give spurious results of the number on the stack is not actually a rational multiple of π. It duplicates the command $\boxed{\to Q\pi}$ on the ALGEBRA menu of the HP-48SX, and has the same name, →Qπ.

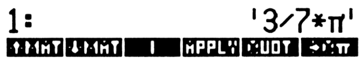
```
« π →NUM / D→Q π * »
```
28S checksum: "25F7"

For example, if the number 1.3463985154 is on the stack,



```
1:        1.34639685154
```

pressing $\boxed{\to Q\pi}$ will return the number '3/7*π'.



```
1:               '3/7*π'
```

(Note that this is read as $\frac{3}{7}\pi$ or $\frac{3\pi}{7}$, not $\frac{3}{7\pi}$.)

Prime Factorization

The following program, called PFACT, gives the prime factorization of the integer in level 1 on the stack. If the integer is already prime, the number itself is returned. For the HP-28S, type the following:

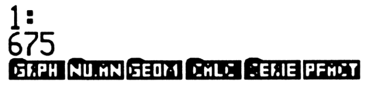
```
« 'N' STO 1 1 CF WHILE 1 FC? REPEAT N √ IP 2 → R K «
WHILE K R ≤ N K MOD 0 ≠ AND REPEAT K 1 + 'K' STO END
'P' * DUP SIZE IF K R > THEN N EXSUB 1 SF ELSE K EXSUB
N K / 'N' STO END » END 'N' PURGE 1 CF »
```
28S checksum: "F129"

For the HP-48SX, the program is slightly different; the symbols not appearing on the keyboard are found as described on p. 52 of the *HP-48SX Owner's Manual.*

```
« DUP 'N' STO 'f' 1 CF WHILE 1 FC? REPEAT N √ IP 2 → R K
« WHILE K R ≤ N K MOD 0 ≠ AND REPEAT K 1 + 'K' STO END
'P' * IF K R > THEN { P N } | 1 SF ELSE { P K } | N K /
'N' STO END » END 'N' PURGE 1 CF »
```
48SX checksum: # 27806d

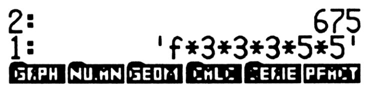
Store the program by typing 'P F A C T $\boxed{\text{STO}}$. As an example of the use of PFACT, suppose you had to reduce the radical $\sqrt{675}$. Type

675



```
1:
675
```

$\boxed{\text{PFACT}}$



```
2:                   675
1:       'f*3*3*3*5*5'
```

to get the prime factorization, which is 3*3*3*5*5. The radical can therefore be reduced by pulling out two factors each of 3 and 5, so that $\sqrt{675} = 15\sqrt{3}$.

Chapter 2. Use in Precalculus Topics

Since the calculus constantly refers to precalculus mathematics, it only makes sense that you should learn to handle precalculus mathematics on the HP calculator in order to do calculus on it. In this chapter we take a brief look at some of the uses in precalculus topics of the HP-28S and HP-48SX.

Algebraic Manipulation

The symbol-manipulating calculators can do quite a bit of algebraic manipulation. Some examples of what can be done are shown starting on p. 110 of the *HP-28S Owner's Manual* and on p. 125 of the *HP-48SX Owner's Manual.* The algebraic manipulation commands are explained in detail in the *HP-28S Reference Manual*, starting on p. 16, and in the *HP-48SX Owner's Manual*, starting on p. 386.

There are some limitations that you should be aware of. Although the calculator will expand and collect by itself, it will not factor by itself. You can walk it through to get simple factoring done, but you must do it "by hand". If you use the program EXCO from the *HP-28S Owner's Manual*, p. 255, or the *HP-48SX Owner's Manual*, p. 568, you are sometimes surprised at what the calculator thinks is simplified form.

Solving Equations

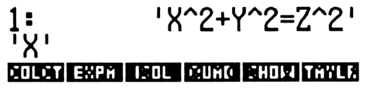
Solving equations is something the calculator can do pretty well. It can do it in either algebraic form, for some equations, or numerical form, for others.

Isolating a Variable

The command $\boxed{\text{ISOL}}$ on the ALGEBRA menu can be used to isolate a single occurrence of a variable in an equation. To use it, enter the equation onto the stack, and then enter the variable you wish to isolate. Then press $\boxed{\text{ISOL}}$.

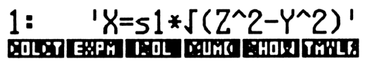
For example, to solve for x in the equation $x^2 + y^2 = z^2$, type the following:

' X ^ 2 + Y ^ 2 = Z ^ 2 $\boxed{\text{ENTER}}$ ' X



```
1:          'X^2+Y^2=Z^2'
'X'
```

$\boxed{\text{ISOL}}$



```
1:    'X=s1*√(Z^2-Y^2)'
```

(on the HP-48SX, ^ is $\boxed{y^x}$). The symbol s1 in the result stands for ±1. That is, the machine gives both solutions to a quadratic equation. More generally, the machine will use De Moivre's Theorem to give you all n solutions of an nth-degree equation. The symbol n1 in a result stands for one of the integers 0, 1, 2, ..., n-1.

For a quadratic equation, the command $\boxed{\text{QUAD}}$ on the SOLV menu also isolates the variable, using the quadratic formula. If there is no other variable in the equation, the result is a numerical answer. If there are other variables, the result is in terms of the other variables.

For example, to solve the equation $x^2 + 3x + 4 = 0$, type in

'X ^ 2 + 3 * X + 4 [ENTER] ' X

```
1:          'X^2+3*X+4'
'X'
COLCT EXPA ISOL QUAD SHOW TAYLR
```

[QUAD].

```
1: 'X=(-3+s1*
   (0,2.64575131106))/
   2'
COLCT EXPA ISOL QUAD SHOW TAYLR
```

The result in this case is given as a pair (s1 = ±1) of complex numbers.

Root Finding

There are at least two methods of finding numerical solutions to an equation in a single variable on the HP-28S and HP-48SX. For a quadratic equation, [QUAD] returns a numerical solution, modulo the symbol s1. Suggestions for further handling quadratic equations are given in the *HP-28S Owner's Manual*, starting on p. 107, and in the *HP-48SX Owner's Manual*, starting on p. 391.

For other equations, the [ROOT] command on the SOLVE menu is used. First enter the equation, then the variable, and then an approximation to the desired root. [ROOT] will find the closest solution to the approximation given.

For example, to find the positive zero of the function $x^3 + 4x^2 -3x -12$, type in

'X ^ 3 + 4 * X ^ 2 - 3 * X - 12
[ENTER] ' X [ENTER] 2

```
2:  'X^3+4*X^2-3*X-12'
1:                 'X'
2
SOLVR ROOT NEW EDEQ STEQ CAT
```

[ROOT].

```
1:        1.73205080757
SOLVR ROOT NEW EDEQ STEQ CAT
```

Other methods of root finding are possible using programs, and will be discussed in the appropriate contexts later.

Evaluating Expressions

[1.4ff]

It is often desired to evaluate an expression involving variables for given values of the variables. The HP Solver environment was created precisely to do that easily. The Solver creates a menu for the expression and for each variable in it, enabling you to easily "plug in" values and get values out.

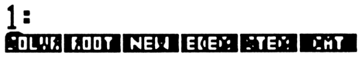
For example, suppose that you wish to evaluate the expression $\sqrt{a^2 + b^2}$ for various values of a and b. First type the expression

' √‾ (A ^ 2 + B ^ 2 [ENTER].

```
1:          '√(A^2+B^2)'
GRAPH NUMAN GEOM CALC SERIE PFMT
```

Then go to the SOLV menu and store the expression as your equation by pressing

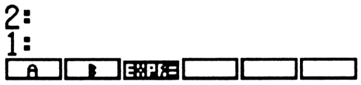
$\boxed{\text{STEQ}}$.



```
1:
```

This command stores the expression in a container named EQ on the USER or VAR menu. Then enter the Solver environment by pressing

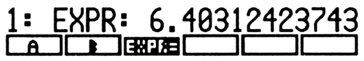
$\boxed{\text{SOLVR}}$.



```
2:
1:
```

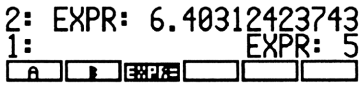
You will see a special menu with choices labeled $\boxed{\text{A}}$, $\boxed{\text{B}}$, and $\boxed{\text{EXPR=}}$. To set A equal to 4, say, and B equal to 5, type

4 $\boxed{\text{A}}$ 5 $\boxed{\text{B}}$. Then press $\boxed{\text{EXPR=}}$ to see the value of the expression.



```
1: EXPR: 6.40312423743
```

To evaluate at A = 4 and B = 3, just type 3 $\boxed{\text{B}}$ and press $\boxed{\text{EXPR=}}$ again.



```
2: EXPR: 6.40312423743
1:               EXPR: 5
```

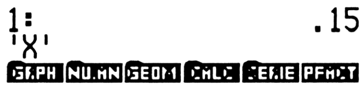
The value of A remains 4 until you change it.

The Solver can also be used to solve for one variable in a formula when the other variables have values given. A good illustration of this is given in the *HP-28S Owner's Manual*, starting on p. 103. The Solver is discussed in the *HP-48SX Owner's Manual* starting on p. 250.

Another way to evaluate an expression is to do by hand what the Solver does for you. First, create a container for each variable in the expression by storing a desired value in each. Then type in the expression, and use $\boxed{\text{EVAL}}$. This procedure is usually used within a program, where the Solver environment is not available.

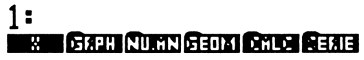
For example, to evaluate the function $f(x) = \frac{\sin x}{x}$ at x = 0.15, type

. 1 5 $\boxed{\text{ENTER}}$ ' X



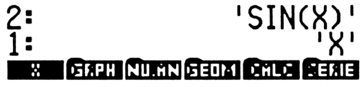
```
1:                   .15
'X'
```

$\boxed{\text{STO}}$



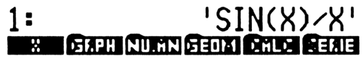
```
1:
```

' S I N (X $\boxed{\text{ENTER}}$ ' X $\boxed{\text{ENTER}}$



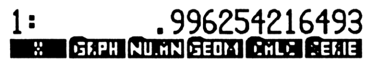
```
2:              'SIN(X)'
1:                   'X'
```

$\boxed{\div}$



```
1:            'SIN(X)/X'
```

$\boxed{\text{EVAL}}$.



```
1:        .996254216493
```

(Make sure the calculator is in radian mode!)

After evaluating an expression, the containers for the variables are left on the USER or VAR menu. It is a good practice to PURGE such containers if they are not going to be used again immediately. Not only it is good housekeeping, but some function commands don't work as you expect when there is a container named for the variable on the USER or VAR menu.

Graphs of Functions

$\boxed{1.4,1.6,4.3,6.1\text{ff}}$

The HP-28S and HP-48SX have many built-in graphics commands that make for flexible and convenient plotting of the graphs of functions. Most of these are accessed through commands on the PLOT menu.

The display on the HP-28S is 137 pixels wide and 32 pixels high. The display on the HP-48SX is 131 pixels wide and 64 pixels high. The display can be thought of as a window showing a portion of the Cartesian plane. The particular portion shown, and therefore its scale, is controlled by the contents of PPAR, the plot parameters.

Plot Parameters

PPAR contains a list, of the form

$$\{\ (x_{min}, y_{min})\ (x_{max}, y_{max})\ X\ n\ (x_{axis}, y_{axis})\ \}.$$

(Two more items are added to this list in the HP-48SX, but we can ignore them for the time being.) The point (x_{min}, y_{min}) is at the lower left corner of the window, and the point (x_{max}, y_{max}) is at the upper right corner. The point (x_{axis}, y_{axis}) is the point at which the axes intersect, and may or may not be visible through the window. If an axis is shown, it will have a tic mark every ten pixels. The variable X is just the independent variable used. The number n is a resolution number, and tells the machine whether you want a pixel turned on in each column of the display.

The default plot parameters for the HP-28S are

{ (-6.8, -1.5) (6.8, 1.6) X 1 (0, 0) }.

For the HP-48SX, the default plot parameters are

{ (-6.5, 3.1) (6.5, 3.2) X 0 (0, 0) FUNCTION Y }.

These parameters put the axes through the center of the screen, with a tic mark every unit. The container PPAR is on the USER or VAR menu; if no such container is there, the DRAW command creates one with the default plot parameters in it.

The contents of PPAR can be edited if you want to change them, or you may use the built-in commands on the PLOT menu. For the HP-28S, the command $\boxed{\text{PMIN}}$ sets the point (x_{min}, y_{min}) to be the point that is in level 1 on the stack . $\boxed{\text{PMAX}}$ does the same for (x_{max}, y_{max}), and $\boxed{\text{AXES}}$, for (x_{axis}, y_{axis}). For the HP-48SX, some of the contents of

PPAR are displayed when you enter the PLOT PLOTR menu. The points (x_{min}, y_{min}) and (x_{max}, y_{max}) are changed using $\boxed{\text{XRNG}}$ and $\boxed{\text{YRNG}}$.

On both machines, the command $\boxed{\text{INDEP}}$ is used for changing the independent variable, and $\boxed{\text{RES}}$ is used to change the resolution number. The command $\boxed{\text{CENTR}}$ adjusts the plot parameters so as to move the window, without changing its length or width, so that the point that is on the stack is shown at the center of the screen. The commands $\boxed{\text{*H}}$ and $\boxed{\text{*W}}$ multiply the height and width of the window by the number on the stack. Other commands are discussed in the *HP-28S Reference Manual*, beginning on p. 152, and in the *HP-48SX Owner's Manual*, beginning on p. 291. The plotting environment of the HP-48SX is much richer, and allows for much greater flexibility.

It is often the case that plot parameters have been adjusted, but now you want the default plot parameters. There are two ways to get them. One is to go to the USER or VAR menu and PURGE the container PPAR. Then when DRAW is used, a new container with the default parameters is created. But that puts the container PPAR first on the menu, which is not usually where you want it.

The other and by far simpler way to get the default plot parameters is push a button. On the HP-48SX, press $\boxed{\text{RESET}}$ on the PLOT PLOTR menu. On the HP-28S, use the following program, called DEFPP (for DEFault Plot Parameters):

```
« { (-6.8,-1.5) (6.8,1.6) X 1 (0,0) } 'PPAR' STO »
                                  28S checksum: "D08E"
```

The advantage of this program is that it is easy to use and it leaves the PPAR container where it is, usually tucked away at the end of the menu. If DEFPP is on the HOME level of the USER menu, they typing D E F P P $\boxed{\text{ENTER}}$ at any time places the default plot parameters on the current level of the USER menu.

Drawing

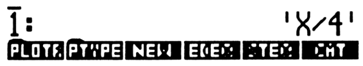
The basic procedure for getting the graph of a function is to type in the function, store it in EQ, and DRAW it. Typing the function is basically the same for both the HP-28S and HP-48SX, although the latter has the additional EquationWriter environment, described starting on p. 226 of the HP-48SX Owner's Manual.

To store the function in EQ, in either machine, enter the PLOT menu and press $\boxed{\text{STEQ}}$. That creates a container named EQ on the USER or VAR menu, if one is not already there, and stores the function in it. The command $\boxed{\text{RCEQ}}$ on the HP-28S recalls the function for you if you want to see what is there; the contents of EQ are shown automatically on the HP-48SX.

Finally, the $\boxed{\text{DRAW}}$ command (press $\boxed{\text{PLOTR}}$ and then $\boxed{\text{DRAW}}$ on the HP-48SX) puts the graph into the display. In the HP-48SX, if you do not first press $\boxed{\text{ERASE}}$, the graph is drawn over whatever is already there.

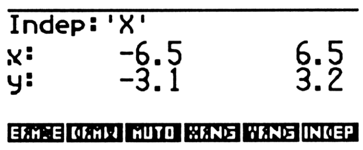
Now let's try an example. Suppose you want to draw the graph of the line $y = \frac{1}{4}x$. Type

' X ÷ 4 ENTER,



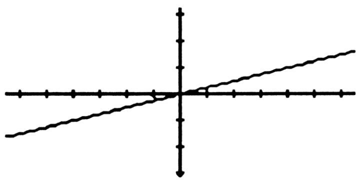
```
1:                'X/4'
```

and then press STEQ to store that expression,



```
Indep: 'X'
x:      -6.5        6.5
y:      -3.1        3.2
```

and finally press DRAW.



Assuming default plot parameters, you will get a nice picture of the line slanting across the screen, passing through the origin as it should.

Press ON (actually ATTN) to get the stack display back again. If you then press DRAW again, the same picture is created again. On the HP-48SX, entering the GRAPH menu recalls the picture immediately.

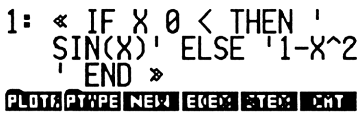
You should now experiment with various functions and plot parameters, to see how the machine behaves, and how the window works.

If you store an equation such as 'SIN(X) = COS(X)' in EQ, the DRAW command plots the graphs of the left-hand side and of the right-hand side simultaneously; this is one way to get two graphs on the screen at the same time.

You may also use a procedure instead of an expression for the function. Suppose you want to create the graph of the function

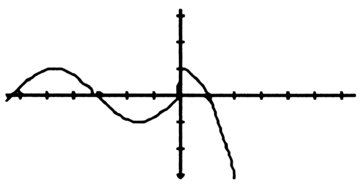
$$f(x) = \begin{cases} \sin x \text{ if } x < 0 \\ 1 - x^2 \text{ if } x \geq 0 \end{cases}.$$

Type « IF X 0 < THEN 'SIN(X)' ELSE '1 - X^2' END ENTER



```
1: « IF X 0 < THEN '
   SIN(X)' ELSE '1-X^2
   ' END »
```

and press STEQ and then DRAW.



Digitizing

If you have adjusted the plot parameters so that the axes do not intersect at the center of the screen, you may have noticed a small cross-hair cursor at the center of the screen . After using the [DRAW] command, the graph is in interactive mode, meaning that the cursor is activated, and can be positioned on any pixel in the display by using the cursor keys. The coordinates of the location of the cursor can be viewed by pressing and holding the [❖] key (the cursor enable key, next to the red key) on the HP-28S, or by pressing [COORD] on the HP-48SX (hide the coordinates by pressing another soft key). The point can be stored on the stack , or "digitized", by pressing the [INS] key on the HP-28S or [ENTER] on the HP-48SX.

Saving

The entire plot that is created by [DRAW] can be stored on the HP-28S by pressing the [DEL] key (next to the [INS] key), which puts onto the stack a binary string representing the display. This string can be stored for future reference; it is just like a snapshot of the screen. This snapshot can be viewed again by putting the binary string back into the display, using the [→LCD] command on the STRING menu. Be aware that the [→LCD] command removes the binary string from the stack; if you want it for future reference without plotting the graph over again, duplicate it and store it first!

Graph storing is automatic on the HP-48SX; graphs are drawn in a portion of memory called PICT, which does not change until you command a change. For example, a new graph is drawn on top of the previous one unless you ERASE the old one first. The contents of PICT can be stored as a graphics object (another data type, equivalent to the binary string representation on the HP-28S) by pressing [PICT] on the PRG DSPL menu and then pressing [RCL]. This graphics object can be viewed again by using the [→LCD] command on the PRG DSPL menu. Also, portions of the display can be selected by setting a mark and referencing the rectangle of which the mark and the cursor are opposite corners; this is described on p. 302 and pp. 337ff of the *HP-48SX Owner's Manual.*

Zooming

The use of the cursor keys and digitizing allows for "zooming in" on a picture with the HP-28S. After plotting a graph, position the cursor at the point that you want to be the upper right corner of the window, press [INS], position the cursor at the point you want to be the lower left corner of the window, press [INS], and then press [ON]. You will see the two points you selected on the stack. To incorporate these points into the plot parameters, press [PMIN] and then [PMAX]. Pressing [DRAW] again will create the enlarged plot you selected.

On the HP-48SX, a ZOOM submenu is built in, described starting on p. 301 of the *HP-48SX Owner's Manual.*

Multiple Graphs

On the HP-48SX, multiple graphs are automatic--just don't erase the previous graph. On the HP-28S, two graphs can be superimposed using the binary strings that represent pictures. Suppose that you have two graphs stored as binary strings under the labels G1 and G2. Press $\boxed{\text{G1}}$ and then $\boxed{\text{G2}}$ to place copies of both binary strings on the stack. Then type in the command OR and press $\boxed{\text{ENTER}}$, or press $\boxed{\text{OR}}$ on the BINARY menu. The two binary strings become one, the union of the previous two in the set-theoretic sense. This string can now be stored and displayed, and combined with other strings to get any number of graphs superimposed.

Zeros by Graphical Methods

With the graph of a function on the display, a zero of the function is the x-coordinate of a point at which the graph crosses the x-axis; such a point is also called an x-intercept of the graph. A zero can therefore be estimated by noting the x-coordinate of an x-intercept. This can be done by positioning the cursor on the x-intercept and reading the coordinates by holding down the cursor-enable key $\boxed{❖}$ or pressing $\boxed{\text{COORD}}$.

Better yet, on the HP-48SX, press $\boxed{\text{FCN}}$ and then $\boxed{\text{ROOT}}$ to get the "exact" value of the zero. To do the same thing on the HP-28S, digitize the x-intercept to get its coordinates onto the stack, and then view the stack by pressing $\boxed{\text{ON}}$. Since it is only the x-coordinate of the x-intercept that we want, turn the point into a pair of real numbers by pressing $\boxed{\text{C→R}}$ on the COMPLEX menu and then $\boxed{\text{DROP}}$ to discard the y-coordinate. Now get the function by pressing $\boxed{\text{RCEQ}}$ on the PLOT menu, and then $\boxed{\text{SWAP}}$. Now type in the variable of the function, usually 'X', and $\boxed{\text{SWAP}}$ again. You should now have the function in level 3, the variable in level 2, and the x-coordinate of the x-intercept in level 1. At this point, press $\boxed{\text{ROOT}}$ on the SOLV menu, and the "exact" value of the zero will be returned.

The process described in the last paragraph for the HP-28S can also be automated. Here is a program called ZERO that does just what was described, starting with the digitized point and returning the zero.

```
      « C→R DROP EQ SWAP 'X' SWAP ROOT »
                                          28S checksum: "26C2"
```

That is, after digitizing and pressing $\boxed{\text{ON}}$ to view the stack, just press $\boxed{\text{ZERO}}$.

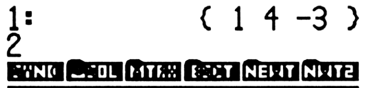
Synthetic Division

One of the major tools for finding zeros of polynomials is synthetic division. Here is a program that takes a list of the coefficients of a polynomial and a multiplier from the stack and returns the quotient and remainder. I call it SYND.

```
« → L M « L L SIZE 0 → N S « 1 N FOR K L K GET S +
DUP M * 'S' STO NEXT → R « N 1 - →LIST M SWAP R » » »
»
                                          28S checksum: "52FC"
                                          48SX checksum: # 30901d
```

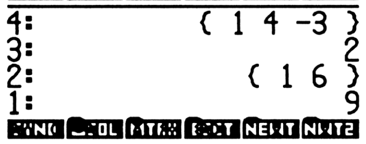
For example, to perform the synthetic division of $x^2 + 4x - 3$ by x - 2, the list of coefficients is { 1 4 -3 } and the multiplier is 2. Therefore type

{ 1 , 4 , 3 $\boxed{\text{CHS}}$ $\boxed{\text{ENTER}}$ 2



```
1:           { 1 4 -3 }
2
```

$\boxed{\text{SYND}}$.



```
4:           { 1 4 -3 }
3:                    2
2:              { 1 6 }
1:                    9
```

The original data as well as the quotient and remainder are returned to the stack; hide the menu bar to see it all on the HP-28S.

Trigonometric Functions

$\boxed{1.6}$

The trigonometric functions are found on the keyboard of the HP-48SX and on the TRIG menu of the HP-28S. Only the functions SIN, COS, and TAN are given there; the others are found by using the fundamental identities,

$$\cot x = \frac{1}{\tan x}, \quad \sec x = \frac{1}{\cos x}, \quad \text{and } \csc x = \frac{1}{\sin x}.$$

Angle Modes

The calculator usually comes set in degree mode, meaning that arguments of trigonometric functions are assumed to be in degrees. To change to radian mode, and have the machine treat arguments of trig functions as radians, press $\boxed{\text{RAD}}$ (on the MODE menu of the HP-28S). The annunciator (2π) or RAD will come on at the top of the display to signal that you are in radian mode.

It is important to be in radian mode for the applications of calculus, including graphs involving the trig functions. To see a demonstration of that fact, try plotting the function sin x in degree mode, and then plot it in radian mode. Type ' S I N (X $\boxed{\text{ENTER}}$ to enter the sine function, press $\boxed{\text{STEQ}}$, and then $\boxed{\text{DRAW}}$. It is a good idea to set your machine in radian mode now, and leave it there.

π

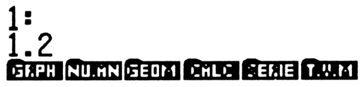
In dealing with radians, the number π will come up a lot. On the HP symbol-manipulating calculators, π and some other numbers are treated as symbolic constants instead of numerical constants. For example, when you press the π key and enter it, the expression 'π' appears on the stack. If you divide by 6, the expression 'π/6' appears on the stack. Then if you press $\boxed{\text{SIN}}$, the expression 'sin(π/6)' appears. To change any of these symbolic expressions into a number, press $\boxed{\to\text{NUM}}$.

An internal flag on the calculator can be cleared to give the numerical rather than symbolic value of π; see p. 206 of the *HP-28S Owner's Manual* or p. 127 of the *HP-48SX Owner's Manual.*

Values

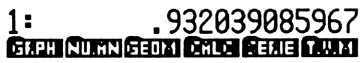
Finding values of the trig functions is as simple as pushing the buttons. To find the sine of 1.2 radians, type

1.2



```
1:
1.2
```

$\boxed{\text{SIN}}$.

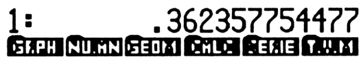


```
1:        .932039085967
```

Similar sequences are used to find the cosine and tangent function values.

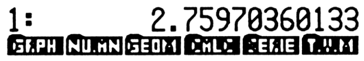
To find the secant of 1.2 radians, type

1.2 $\boxed{\text{COS}}$



```
1:        .362357754477
```

and then press the $\boxed{1/x}$ key.



```
1:        2.75970360133
```

Values of the cosecant and cotangent function are found similarly.

Inverse Trig Functions

$\boxed{6.6}$

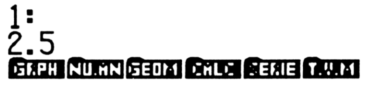
Values of the inverse trig functions arcsine, arccosine, and arctangent are found by pressing the buttons $\boxed{\text{ASIN}}$, $\boxed{\text{ACOS}}$, and $\boxed{\text{ATAN}}$ (on the TRIG menu of the HP-28S). The arcsine and arctangent functions always return a number in the range from $-\pi/2$ to $\pi/2$, and the arccosine function always returns a value between 0 and π. These are the principal value ranges. (There is a discussion of principal branches in the context of complex function theory in the *HP-28S Reference Manual*, but it is of little help until you have had a course in complex variables.)

If you want values of the arccotangent, use the formula

$$\text{arccot } x = \begin{cases} \arctan \frac{1}{x} \text{ if } x > 0 \\ \frac{\pi}{2} + \arctan \frac{1}{x} \text{ if } x < 0 \end{cases}$$

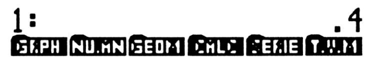
For example, to find the arccotangent of 2.5, type

2.5,



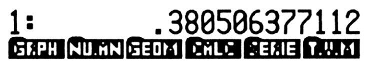
```
1:
2.5
```

press the $\boxed{1/x}$ key,



```
1:                   .4
```

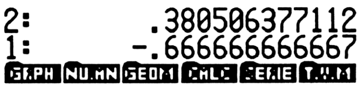
and then press $\boxed{\text{ATAN}}$.



```
1:        .380506377112
```

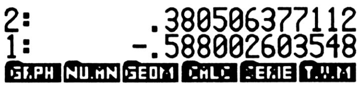
To find the arccotangent of -1.5, type

1.5 $\boxed{\text{CHS}}$ $\boxed{\text{1/x}}$



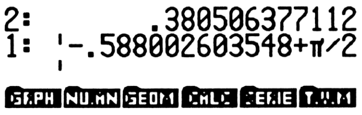
```
2:        .380506377112
1:       -.666666666667
```

$\boxed{\text{ATAN}}$



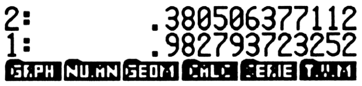
```
2:        .380506377112
1:       -.588002603548
```

π $\boxed{\text{ENTER}}$ $2 \div +$



```
2:        .380506377112
1: '-.588002603548+π/2
```

$\boxed{\to\text{NUM}}$.



```
2:        .380506377112
1:        .982793723252
```

($\boxed{\text{CHS}}$ is replaced by $\boxed{+/-}$ on the HP-48SX; also, no $\boxed{\text{ENTER}}$ is necessary after typing π.) These cases must be taken into account in order to get the proper principal value range.

For values of the arcsecant function, use the formula

$$\text{arcsec } x = \begin{cases} \arccos \frac{1}{x} \text{ if } x > 0 \\ -\arccos \frac{1}{x} \text{ if } x < 0 \end{cases}$$

For values of the arccosecant function, use the formula

$$\text{arccsc } x = \begin{cases} \arcsin \frac{1}{x} \text{ if } x > 0 \\ -\pi - \arcsin \frac{1}{x} \text{ if } x < 0 \end{cases}$$

It may be that the number on the stack is the sine, cosine, or tangent of a rational multiple of π. The program « ASIN →Qπ SIN », which I call →SQ, will take the decimal number on the stack and express it as the sine of a rational multiple of π. The program « ACOS →Qπ COS », called →CQ, will express it as the cosine of a rational multiple of π. The number on the stack must be between -1 and 1 in order for these programs to work. The program « ATAN →Qπ TAN », called →TQ, will express the decimal number on the stack as the tangent of a rational multiple of π. If the number on the stack is not actually the sine, cosine, or tangent of a rational multiple of π, you will get meaningless results. Your calculator must also be in radian mode for these programs to be valid.

Applications of Trigonometry

Formulas such as the Law of Sines and the Law of Cosines can be placed into the Solver for efficient handling in the solution of triangles and other applications of the trigonometric functions. Vectors, another topic that often comes up in trigonometry, can be handled directly, as we discuss below under the heading of Linear Algebra.

The solution of triangles can also be automated. Here are four programs that handle the different cases:

ASA: given two angles and the included side
SAS: given two sides and the included angle
SSS: given the three sides
SSA: given two sides and the angle opposite one of them (this is the only ambiguous case, and there may be one solution, two solutions, or no solution at all)

It is recommended that these programs be placed in a TSOL (for Triangle SOLutions) subdirectory, whose menu would look like

ASA SAS SSS SSA QUIT

• QUIT is the program « UP » or « UPDIR ».

• SSA is the program

```
« → a b A « b A SIN * → H « a H IF < THEN " No Triangle"
ELSE 'a(given)' a = 'b(given)' b = 'A(given)' A = 60 IF FS? THEN π
→NUM ELSE 180 END → S « a H IF == THEN S 2 / 'B' SWAP =
S 2 / A - 'C' SWAP = '√(b^2-a^2)' EVAL 'c' SWAP = ELSE a b IF
≥ THEN b A SIN * a / ASIN DUP 'B' SWAP = SWAP A + S
SWAP - DUP 'C' SWAP = SWAP SIN a * A SIN / 'c' SWAP =
ELSE b A SIN * a / ASIN → B « B S B - R→C 'B' SWAP = S
A B + - B A - → C C1 « C C1 R→C 'C' SWAP = a C SIN * A
SIN / a C1 SIN * A SIN / R→C 'c' SWAP = » » END END » END
» » »
```

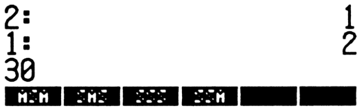
28S checksum: "2C9A"

For the HP-48SX, the number 60 in the second line of this program should be changed to the number -17. With that change, the 48SX checksum is # 46041d.

To use this program, enter the two sides and the angle onto the stack in the order S, S, A. Make sure that you are using the same angle mode throughout. Then press $\boxed{\text{SSA}}$. The labeled sides and angles will be left on the stack. If there are two solutions, the answers are given as ordered pairs; the first entries constitute one solution, and the second entries, another.

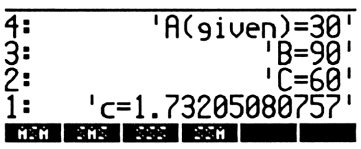
For example, given the sides a = 1, b = 2, and angle A = 30°, put the calculator into degree mode and type

1 $\boxed{\text{ENTER}}$ 2 $\boxed{\text{ENTER}}$ 30



```
2:                1
1:                2
30
```

$\boxed{\text{SSA}}$.



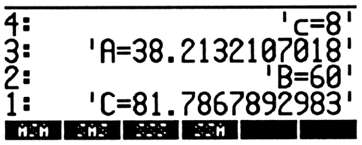
```
4:      'A(given)=30'
3:             'B=90'
2:             'C=60'
1:  'c=1.73205080757'
```

• SSS is the program

```
« → a b c « 'a' a = 'b' b = 'c' c = '(b^2+c^2-a^2)/(2*b*c)' EVAL
ACOS 'A' SWAP = '(a^2+c^2-b^2)/(2*a*c)' EVAL ACOS 'B' SWAP =
'(a^2+b^2-c^2)/(2*a*b)' EVAL ACOS 'C' SWAP = » »
                                        28S checksum: "54BE"
                                        48SX checksum: #54546d
```

To use this program, enter the lengths of the three sides onto the stack and press $\boxed{\text{SSS}}$. For example, if the given sides are 5, 7, and 8, type

5 $\boxed{\text{ENTER}}$ 7 $\boxed{\text{ENTER}}$ 8 $\boxed{\text{SSS}}$.



```
4:              'c=8'
3:   'A=38.2132107018'
2:             'B=60'
1:   'C=81.7867892983'
```

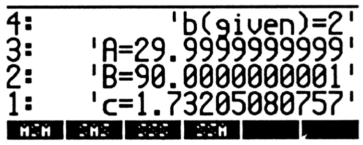
• SAS is the program

```
« → a C b « IF a b > THEN b a 'b' STO 'a' STO END 'a(given)'
a = 'C(given)' C = 'b(given)' b = '√(a^2+b^2-2*a*b*COS(C))' EVAL → c
« a c / C SIN * ASIN DUP 'A' SWAP = SWAP C + 60 IF FS?
THEN π →NUM ELSE 180 END SWAP - 'B' SWAP = 'c' c = » » »
                                        28S checksum: "834B"
```

For the HP-48SX, the number 60 in the third line of this program should be changed to the number -17. With that change, the 48SX checksum is # 46305d.

To use this program, enter the sides and the included angle in the order S, A, S. Then press $\boxed{\text{SAS}}$. Again, be consistent in the angle mode used. For example, if the two sides are 1 and 2 and the included angle is 60°, put the calculator into degree mode and type

1 $\boxed{\text{ENTER}}$ 60 $\boxed{\text{ENTER}}$ 2 $\boxed{\text{SAS}}$.



```
4:       'b(given)=2'
3:   'A=29.9999999999'
2:   'B=90.0000000001'
1:   'c=1.73205080757'
```

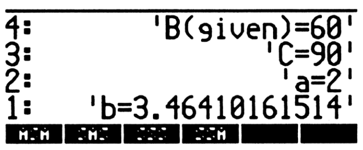
• ASA is the program

```
« → A c B « 'A(given)' A = 'c(given)' c = 'B(given)' b = 60 IF FS?
THEN π →NUM ELSE 180 END A B + - → C « 'C' C = A SIN c
* C SIN / 'a' SWAP = B SIN c * C SIN / 'b' SWAP = » » »
                                        28S checksum: "3F25"
```

For the HP-48SX, the number 60 in the first line of this program should be changed to the number -17. With that change, the 48SX checksum is # 13387d.

To use this program, enter the angles and the included side in the order A, S, A. Be consistent with the angle mode. For example, if the angles given are 30° and 60° and the included side is 4, then type

30 $\boxed{\text{ENTER}}$ 4 $\boxed{\text{ENTER}}$ 60 $\boxed{\text{ASA}}$.



```
4:       'B(given)=60'
3:              'C=90'
2:               'a=2'
1:   'b=3.46410161514'
```

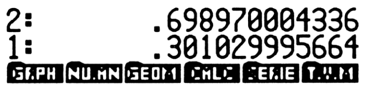
Logarithmic and Exponential Functions

$\boxed{6.1,6.2,6.3}$

The $\boxed{\text{LOG}}$ command (on the LOGS menu of the HP-28S) returns the common logarithm of the number in level 1 on the stack. That is, if x is on the stack, $\boxed{\text{LOG}}$ returns $\log_{10}$ x. $\boxed{\text{ALOG}}$ is the common antilogarithm, the same as $\boxed{10^x}$. $\boxed{\text{LN}}$ is the natural logarithm, or logarithm base e, and $\boxed{\text{EXP}}$ is the natural exponential function, the same as $\boxed{e^x}$. Other commands on the LOGS menu are discussed in the *HP-28S Reference Manual*, beginning on p. 133.

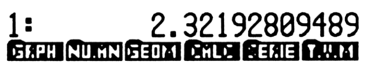
Logarithms with any positive base $b \neq 1$ can be found from the formula $\log_b x = \frac{\log x}{\log b}$. For example, to find $\log_2 5$, type

5 $\boxed{\text{LOG}}$ 2 $\boxed{\text{LOG}}$



```
2:      .698970004336
1:      .301029995664
```

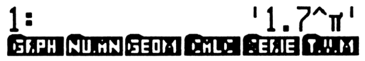
÷.



```
1:      2.32192809489
```

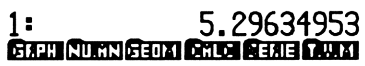
Exponentiation with any base b > 0 is easily accomplished with the $\boxed{\wedge}$ or $\boxed{y^x}$ command. For example, to find $(1.7)^\pi$, type

1.7 $\boxed{\text{ENTER}}$ $\boxed{\pi}$ ^



```
1:             '1.7^π'
```

$\boxed{\to\text{NUM}}$.



```
1:         5.29634953
```

Linear Algebra

Linear algebra is the broad area covering systems of linear equations, matrices, determinants, and vectors and vector spaces. At the precalculus level, everything we need to do can be handled very efficiently by the built-in functions of the calculator, with the exception of the Gauss-Jordan reduction technique.

Vectors

$\boxed{12.2}$

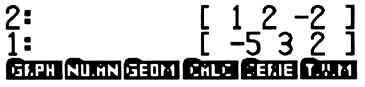
A vector in the HP-28S or HP-48SX is a set of numbers (coordinates) enclosed in brackets, [ ]. To enter a vector, start with $\boxed{\text{[}}$ on the HP-28S or $\boxed{\text{[ ]}}$ on the HP-48SX, and then enter the numbers, typing either a space or a comma between coordinates.

For example, to enter the vector [3 4 -1], type [3 , 4 , 1 $\boxed{\text{CHS}}$ $\boxed{\text{ENTER}}$.

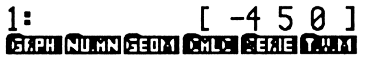
Vector addition and subtraction is accomplished using the $\boxed{+}$ and $\boxed{-}$ keys, just as with real numbers. However, if the vectors do not have the same length (number of coordinates), you will get an error message.

For example, to find the sum of the two vectors [1 2 -2] and [-5 3 2], type

[1 , 2 , 2 $\boxed{\text{CHS}}$ $\boxed{\text{ENTER}}$ [5 $\boxed{\text{CHS}}$, 3 , 2 $\boxed{\text{ENTER}}$


```
2:           [ 1 2 -2 ]
1:           [ -5 3 2 ]
```

$\boxed{+}$.


```
1:           [ -4 5 0 ]
```

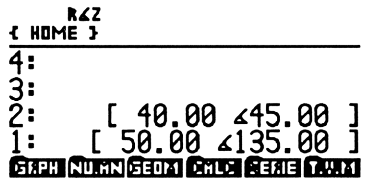
The scalar product of a number and a vector can be found by using the $\boxed{\times}$ key. Just place the vector and the scalar in levels 1 and 2 (in either order) and multiply.

Often in elementary physical applications, vectors are given in polar form, rather than rectangular form. For example, you may be asked to find the resultant of the two forces, 40 pounds in the direction of 45°, and 50 pounds in the direction of 135°. In each case, the vector is given in polar form. In order to add them, it is necessary first to translate them to rectangular form.

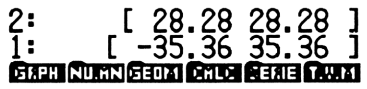
Since the angles in this example are given in degrees, it is easiest to switch the calculator to degree mode. Then enter the vectors and translate them, adding and retranslating. On the HP-28S, type the following: [40 , 45 $\boxed{\text{ENTER}}$ $\boxed{\text{P→R}}$ [50 , 135 $\boxed{\text{ENTER}}$ $\boxed{\text{P→R}}$ $\boxed{+}$ $\boxed{\text{R→P}}$. Switch to 2 FIX to read the result. The commands $\boxed{\text{P→R}}$ and $\boxed{\text{R→P}}$ are found on the TRIG menu.

On the HP-48SX, first press $\boxed{\text{POLAR}}$ to enter polar mode, and then type

40 $\boxed{\text{ENTER}}$ 45 $\boxed{\text{2D}}$ 50 $\boxed{\text{ENTER}}$ 135 $\boxed{\text{2D}}$


```
R∡Z
{ HOME }
4:
3:
2:     [ 40.00 ∡45.00 ]
1:    [ 50.00 ∡135.00 ]
```

$\boxed{\text{POLAR}}$


```
2:     [ 28.28 28.28 ]
1:    [ -35.36 35.36 ]
```

+

```
1:          [ -7.07 63.64 ]
GRPH NU.MN GEOM CALC SERIE T.V.M
```

$\boxed{\text{POLAR}}$.

```
1:        [ 64.03 ∡96.34 ]
GRPH NU.MN GEOM CALC SERIE T.V.M
```

The resulting vector [64.03 96.34] represents a force of 64.03 pounds in the direction of 96.34°.

The length of a vector is given by the $\boxed{\text{SIZE}}$ command on the ARRAY menu of the HP-28S and on the PRG OBJ menu of the HP-48SX. The magnitude or norm or absolute value of a vector is given by the $\boxed{\text{ABS}}$ command on the ARRAY or MTH VECTR menu.

$\boxed{\boxed{\text{12.3}}}$

Dot and cross products of vectors are also defined in the machine. The $\boxed{\text{DOT}}$ command computes the dot product of any two vectors of the same length. The $\boxed{\text{CROSS}}$ command computes the vector that is the cross product of two vectors of length three. These will be quite useful to us later on.

One restriction that is sometimes unhandy is that the entries of a vector must be real numbers or complex numbers. One cannot use symbols as the coordinates of a vector. If you wish to do so, you might try using lists, whose entries can be anything, and create routines to do arithmetic and whatever else you wish with them.

Another way to enter a vector is to place the coordinates of the vector on the stack, place the number of coordinates in level 1, and then use the $\boxed{\to\text{ARRY}}$ command on the ARRAY or PRG OBJ menu. For example, to create the vector [3 5 2 1], you may type

3 $\boxed{\text{ENTER}}$ 5 $\boxed{\text{ENTER}}$ 2 $\boxed{\text{ENTER}}$ 1 $\boxed{\text{ENTER}}$ 4

```
3:                      5
2:                      2
1:                      1
4
OBJ→ EQ→ →ARR →LIST →STR →TAG
```

$\boxed{\to\text{ARRY}}$.

```
1:              [ 3 5 2 1 ]
OBJ→ EQ→ →ARR →LIST →STR →TAG
```

This is the method most often used inside a program.

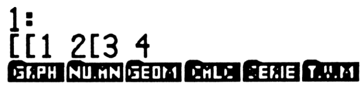
Matrices and Determinants

$\boxed{\boxed{\text{APPENDIX A}}}$

A matrix is perhaps best regarded as a vector of vectors. To enter a matrix on the HP-28S, start with two brackets, [[. Again separate entries by spaces or commas, and signal the beginning of a new row by typing another [. Upon entry, the machine will supply all the missing brackets.

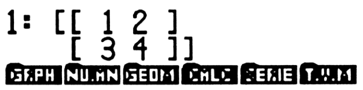
For example, to enter the matrix $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$, type

[[1,2[3,4



```
1:
[[1 2[3 4
```

ENTER .



```
1: [[ 1 2 ]
    [ 3 4 ]]
```

On the HP-48SX, use the MatrixWriter environment described on p. 346 of the *HP-48SX Owner's Manual.* To enter the same matrix as above, type MATRIX 1 SPC 2 ENTER ▼ 3 SPC 4 ENTER ENTER .

Arithmetic with matrices is also done using the regular arithmetic keys. As with vectors, the matrices must have the same size (same numbers of rows and of columns) in order to form sums and differences. Matrices can also be multiplied by using the × key, but they must be conformable. That is, the number of columns of the matrix in level 2 must be the same as the number of rows of the matrix in level 1.

The product of a scalar and a matrix can be found by using the × key. Just place the matrix and the scalar in levels 1 and 2 (in either order) and multiply.

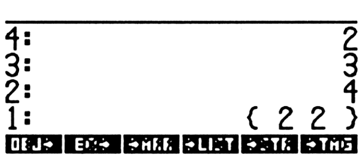
The SIZE command, when applied to a matrix, returns a list in which the first element is the number of rows and the second element is the number of columns. If a matrix is square (has the same number of rows as columns), then the DET command on the ARRAY menu of the HP-28S and on the MTH MATR menu of the HP-48SX returns the determinant of the matrix. If the determinant of a square matrix is not zero, then the inverse of the matrix exists, and the inverse can be found by using the 1/x command.

(*Warning*: On early versions of the HP-48SX, there may be a bug in the INV or 1/x command that causes trouble when inverting a matrix larger than 7 by 7. For large matrices, put an identity matrix of the same size on level 2 by using IDN , put the matrix to be inverted on level 1, and use ÷ .)

Other commands on the ARRAY menu are described in the *HP-28S Reference Manual,* beginning on p. 63. Other commands on the MTH MATR menu are described on p. 359 of the *HP-48SX Owner's Manual.*

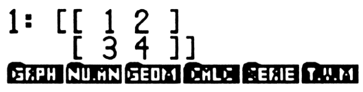
A matrix can also be entered by placing its entries on the stack, and then using a list of two numbers to specify the size of the matrix and using →ARRY . For example, the enter the matrix $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$, we can type

1 ENTER 2 ENTER 3 ENTER 4 ENTER
{2,2



```
4:              2
3:              3
2:              4
1:        { 2 2 }
```

→ARRY .



```
1: [[ 1 2 ]
    [ 3 4 ]]
```

This is most often done in the midst of a program.

A matrix is most easily edited on the HP-28S by using the EDIT command. With the matrix on level 1, type EDIT, and then make whatever changes are necessary. On the HP-48SX, use the MatrixWriter. With the matrix on level 1, press ▼, and use the cursor keys to move to the desired entries and enter new numbers. In both cases, to save the edited result, press ENTER; to discard the edited version, press ON; in either case, the matrix appears again on level 1.

Systems of Linear Equations

APPENDIX B

The built-in matrix functions of the HP calculators enable the solving of systems of linear equations. Each such system can be represented as a matrix equation of the form **AX** = **B**, with **A** the coefficient matrix, **X** the column of unknowns, and **B** the column of constants. If **A** is invertible, then $\mathbf{X} = \mathbf{A}^{-1}\mathbf{B}$, and $\mathbf{A}^{-1}\mathbf{B}$ can be found by placing the vector **B** in level 2, matrix **A** in level 1, and pressing ÷. Even if **A** is not a square matrix, the same approach can be used, as is discussed in the *HP-28S Reference Manual* on p.s 66-67 and on p. 362 of the *HP-48SX Owner's Manual.* The use of ÷ involves some roundoff error, typically.

Gauss-Jordan Reduction

The Gauss-Jordan reduction method of solving a system of linear equations begins with the augmented matrix [**A B**] of the system **AX** = **B**. Then by using elementary row operations, the matrix is transformed to reduced echelon form, from which the solution of the original system can be easily read.

The three elementary row operations are (1) interchanging two rows, (2) multiplying a row vector by a nonzero scalar, and (3) replacing a row vector by the vector sum of that row and another row. The latter two can be combined and used repeatedly to change the column containing a chosen nonzero element into a column of zeros, except that the chosen element will be replaced by a 1. This process is called pivoting, and the chosen element is called the *pivot.* The Gauss-Jordan process simply pivots on diagonal elements, insofar as that is possible, until the reduced-echelon form of the matrix is obtained.

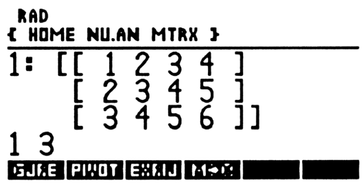
In the pivoting process, an entry often is not "zeroed out" as desired, but replaced by a very small number, such as 1.2×10^{-11}. This is due to round-off error in the calculator. To overcome the problem, we use the program N.DEC (or just RND on the HP-48SX) to instruct the calculator how many decimal places to take seriously.

Here is a program called EXRIJ for exchanging rows I and J of a matrix. It assumes that the matrix is in level 3 of the stack, I is in level 2, and J is in level 1. The matrix, with rows I and J interchanged, is returned to level 1. Be careful, as you type in the following program, to distinguish between 0 (zero) and the letter capital O. Also be careful to observe the spacing given in this program (and all others, for that matter).

```
« → I J « DUP SIZE 1 GET IDN 'E' STO 0 0 'E(I,I)' STO 'E(J,J)'
STO 1 1 'E(I,J)' STO 'E(J,I)' STO E SWAP * 'E' PURGE » »
                                        28S checksum: "E2F8"
                                        48SX checksum: # 6505d
```

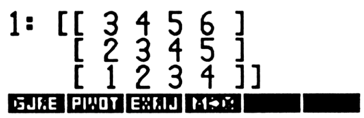
For example, to interchange rows 1 and 3 of the matrix on the stack in level 1, type

1 $\boxed{\text{SPC}}$ 3



```
RAD
{ HOME NU.AN MTRX }
1: [[ 1 2 3 4 ]
    [ 2 3 4 5 ]
    [ 3 4 5 6 ]]
1 3
```

$\boxed{\text{EXRIJ}}$.



```
1: [[ 3 4 5 6 ]
    [ 2 3 4 5 ]
    [ 1 2 3 4 ]]
```

The next program is called PIVOT, and is used to pivot on the element in row K and column L. (The specified element cannot be zero.) It assumes the matrix is in level 3, K is in level 2, and L is in level 1. For the HP-28S, the program is

```
« → A K L « A SIZE 1 GET → M « M IDN 'P' STO 1 M FOR I
'A(I,L)' EVAL 'P(I,K)' STO NEXT P INV A * 9 N.DEC 'P' PURGE » »
»
```

28S checksum: "BF81"

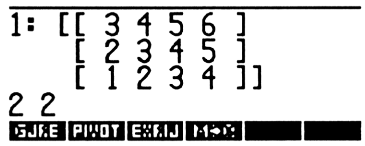
For the HP-48SX, the program is

```
« → A K L « A SIZE 1 GET → M « M IDN 'P' STO 1 M FOR I
'A(I,L)' EVAL 'P(I,K)' STO NEXT M IDN P / A * 9 RND 'P' PURGE
» » »
```

48SX checksum: # 58129d

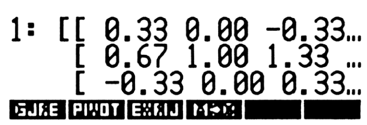
For example, to pivot on the 2,2 element of the matrix on the stack, type

2 $\boxed{\text{SPC}}$ 2



```
1: [[ 3 4 5 6 ]
    [ 2 3 4 5 ]
    [ 1 2 3 4 ]]
2 2
```

$\boxed{\text{PIVOT}}$.



```
1: [[ 0.33 0.00 -0.33…
    [ 0.67 1.00 1.33 …
    [ -0.33 0.00 0.33…
```

EXRIJ and PIVOT are enough now to row-reduce a matrix. You can use them to row-reduce a matrix "by hand" by simply pivoting on the diagonal elements of a matrix, interchanging rows when it is necessary because of the appearance of zeros on the diagonal.

The following program, called GJRED, automates the row-reduction process, starting with a matrix and returning both the matrix and the reduced-echelon form. It also uses the largest possible element to pivot on ("partial column pivoting"), thereby reducing roundoff error as much as possible. The beep at the end is just to let you know when it is done.

```
« 'A' STO A A SIZE LIST→ DROP 'N' STO 'M' STO 1 1 'C' STO
'R' STO 1 CF WHILE 1 FC? REPEAT 'A(R,C)' EVAL ABS 'T' STO R
'K' STO IF R M < THEN 1 R + M FOR I IF 'A(I,C)' EVAL ABS
DUP 'T1' STO T > THEN T1 'T' STO I 'K' STO END NEXT END IF
T 0 > THEN IF K R ≠ THEN A R K EXRIJ 'A' STO END A R C
PIVOT 'A' STO 1 'R' STO+ IF R M > THEN 1 SF END END 1 'C'
STO+ IF C N > THEN 1 SF END END A { T T1 K R C M N A
} PURGE 660 .4 BEEP 1 CF »
                                        28S checksum: "E59F"
                                        48SX checksum: # 14119d
```

The command STO+ is found on the STORE menu of the HP-28S and on the *right-shifted* MEMORY menu of the HP-48SX. For the HP-48SX, the command LIST→ must be replaced by OBJ→.

Often, especially when solving a system of linear equations, you want to see the results of the Gauss-Jordan reduction in fraction form. The last column of the reduced matrix contains the solution, if the solution is unique. If the solution is not unique, then a row-by-row inspection of the reduced matrix is necessary to specify the complete solution. Each entry of the solution could be isolated and have →Q applied to it, but here is a program that allows for an entire matrix to be transformed into fraction form at once. The program is called M→Q, and takes the matrix from level 2 and returns lists of the elements, row by row, in fraction form.

```
« → A « A SIZE LIST→ DROP → M N « 1 M FOR I 1 N FOR J
'A(I,J)' EVAL D→Q NEXT N →LIST NEXT » » »
                                        28S checksum: "114F"
```

Unfortunately, M→Q may not work properly on the HP-48SX when the command D→Q is replaced by →Q, because →Q uses more precision than is left after PIVOT rounds to 9 decimal places. To make it work, first install the following version of D→Q:

```
« RCLF 8 FIX SWAP →Q SWAP STOF »
                                        48SX checksum: # 6145d
```

This program limits the precision of the →Q command, as described on p. 136 of the *HP-48SX Owner's Manual.* Now the program M→Q will work, except that LIST→ must be replaced by OBJ→. With that change, the 48SX checksum is # 31133d.

Combinatorics

There are several functions built into HP calculators that calculate certain combinatoric numbers. These include the factorials, permutations and combinations. On the HP-28S, the commands are $\boxed{\text{FACT}}$ on the REAL menu and $\boxed{\text{COMB}}$ and $\boxed{\text{PERM}}$ on the STAT menu. On the HP-48SX, the commands are $\boxed{\ !\ }$, $\boxed{\text{COMB}}$, and $\boxed{\text{PERM}}$, all on the MTH PROB menu.

To find n!, put n on the stack and press $\boxed{\text{FACT}}$ or $\boxed{\ !\ }$. If the number x on the stack is not a positive integer, the calculator returns the gamma function $\Gamma(x + 1)$.

To find $\binom{n}{k}$, the number of combinations of n things taken k at a time, put n and k on the stack and press $\boxed{\text{COMB}}$. For example, to compute $\binom{5}{2}$, type 5 $\boxed{\text{ENTER}}$ 2 $\boxed{\text{COMB}}$.

To find P(n,k), the number of permutations of n things taken k at a time, put n and k on the stack and press $\boxed{\text{PERM}}$. For example, to compute P(5,2), type 5 $\boxed{\text{ENTER}}$ 2 $\boxed{\text{PERM}}$.

These can be used together to compute probabilities, etc. For example, to compute the probability of a full house in a random draw of five cards from a standard playing deck, which is

$$\frac{13\cdot\binom{4}{3}\cdot 12\cdot\binom{4}{2}}{\binom{52}{5}},$$

type 13 $\boxed{\text{ENTER}}$ 4 $\boxed{\text{ENTER}}$ 3$\boxed{\text{COMB}}$ $\boxed{\times}$ 12 $\boxed{\times}$ 4 $\boxed{\text{ENTER}}$ 2 $\boxed{\text{COMB}}$ $\boxed{\times}$ 52 $\boxed{\text{ENTER}}$ 5 $\boxed{\text{COMB}}$ $\boxed{\div}$. Use →Q to see the fraction form.

Elementary Plane Analytic Geometry

Many of the operations of analytic geometry are easy to do on the HP-28S and HP-48SX because of the ability of the calculators to handle points (they think they are handling complex numbers) and vectors. Here are some tricks and little programs.

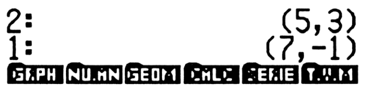
Distance Formula

$\boxed{\boxed{1.2}}$

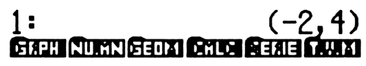
To find the distance between points (a, b) and (c, d) in the plane, just find the absolute value of the difference of the complex numbers (a, b) and (c, d).

For example, to find the distance between (5, 3) and (7, -1), type

(5 , 3 $\boxed{\text{ENTER}}$ (7 , 1 $\boxed{\text{CHS}}$ $\boxed{\text{ENTER}}$

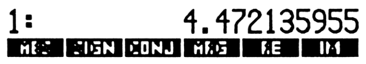


```
2:                (5,3)
1:               (7,-1)
```



```
1:               (-2,4)
```

$\boxed{\text{ABS}}$.



```
1:          4.472135955
```

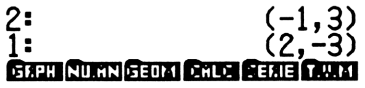
Remember that $\boxed{\text{CHS}}$ is replaced by $\boxed{+/-}$ on the HP-48SX. You may also type $\boxed{\text{SPC}}$ instead of the comma.

Midpoint Formula

To find the midpoint of the segment whose endpoints are (a, b) and (c, d) in the plane, just find the average of the complex numbers (a, b) and (c, d).

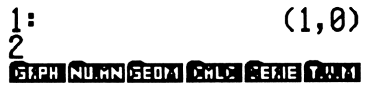
For example, to find the midpoint between (-1, 3) and (2, -3), type

(1 $\boxed{\text{CHS}}$, 3 $\boxed{\text{ENTER}}$ (2 , 3 $\boxed{\text{CHS}}$ $\boxed{\text{ENTER}}$



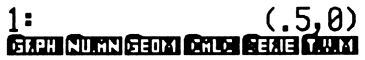
```
2:                (-1,3)
1:                (2,-3)
```

+ 2



```
1:                 (1,0)
2
```

÷.



```
1:                (.5,0)
```

The Line on Two Points

$\boxed{\boxed{1.2}}$

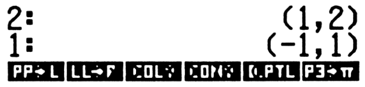
This program , called PP→L, takes two points (in complex number form) from the stack and returns the equation of the line they determine.

```
« → P1 P2 « P1 C→R 1 3 →ARRY P2 C→R 1 3 →ARRY CROSS →
L « L 1 GET 'X' * L 2 GET 'Y' * + L 3 GET + 0 = » » »
```

28S checksum: "4268"
48SX checksum: # 46969d

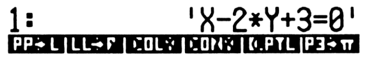
For example, to get the line on the points (1, 2) and (-1, 1), type

(1 , 2 $\boxed{\text{ENTER}}$ (1 $\boxed{\text{CHS}}$, 1 $\boxed{\text{ENTER}}$



```
2:                 (1,2)
1:                (-1,1)
```

$\boxed{\text{PP→L}}$.



```
1:           'X-2*Y+3=0'
```

The name of the program reminds you to enter two points, and the output is a line.

The Point on Two Lines

This program, called LL→P, takes two vectors from the stack, representing two lines in the plane, and returns the point of intersection of the two lines. The line ax + by + c = 0 is entered as the vector [a b c]. If the two lines are parallel, the result "PARALLEL" is returned.

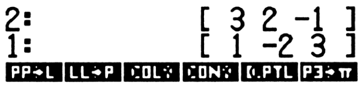
```
« CROSS ARRY→ DROP → A B C « IF C 0 == THEN "PARALLEL"
ELSE A C / B C / R→C END » »
```

28S checksum: "9900"
48SX checksum: # 27494d

On the HP-48SX, the command ARRY→ must be replaced with OBJ→.

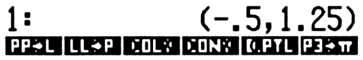
For example, to find the point of intersection of the two lines 3x + 2y - 1 = 0 and x - 2y + 3 = 0, enter the coefficients as vectors:

[3 , 2 , 1 $\boxed{\text{CHS}}$ $\boxed{\text{ENTER}}$ [1 , 2 $\boxed{\text{CHS}}$, 3 $\boxed{\text{ENTER}}$



```
2:          [ 3 2 -1 ]
1:          [ 1 -2 3 ]
```

$\boxed{\text{LL→P}}$.



```
1:        (-.5,1.25)
```

The name of the program reminds you to enter two lines, and the result is a point.

Collinear Points

This program, called COL?, takes three points from the stack and determines whether they are collinear. If so, the line on which they lie is given. Here is the program for the HP-28S:

```
« PP→L 'L' STO C→R 'Y' STO 'X' STO L EVAL DUP 1 EXGET
SWAP 3 EXGET IF == THEN "YES" L ELSE "NO" END { L X Y }
PURGE »
```

28S checksum: "BC0B"

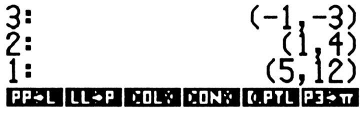
Here is the program for the HP-48SX:

```
« PP→L 'L' STO C→R 'Y' STO 'X' STO L EVAL OBJ→ DROP2 IF ==
THEN "YES" L ELSE "NO" END { L X Y } PURGE »
```

48SX checksum: # 17469d

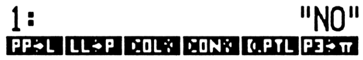
For example, to determine whether the points (-1, -3), (1, 4), and (5, 12) are collinear, type

(1 $\boxed{\text{CHS}}$, 3 $\boxed{\text{CHS}}$ $\boxed{\text{ENTER}}$ (1 , 4 $\boxed{\text{ENTER}}$ (5 , 12 $\boxed{\text{ENTER}}$



```
3:            (-1,-3)
2:              (1,4)
1:             (5,12)
```

$\boxed{\text{COL?}}$.



```
1:               "NO"
```

Then try the points (-1, -3), (1, 4), and (5, 18).

Concurrent Lines

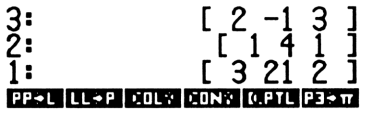
This program, called CON?, takes three vectors, representing three lines in the plane, from the stack and determines whether the lines all pass through the same point. If so, the point of intersection is given. The line ax + by + c = 0 is entered as the vector [a b c].

« → L1 L2 « L1 L2 CROSS DOT IF 0 == THEN "YES" L1 L2 LL→P ELSE "NO" END» »

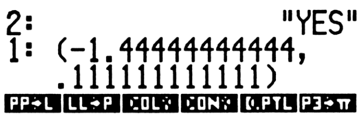
28S checksum: "A102"
48SX checksum: # 54109d

For example, to determine whether the lines 2x - y + 3 = 0, x + 4y + 1 = 0, and 3x + 21y + 2 = 0 are concurrent, type

[2 , 1 CHS , 3 ENTER [1 , 4 , 1 ENTER
[3 , 21 , 2 ENTER


```
3:          [ 2 -1 3 ]
2:           [ 1 4 1 ]
1:          [ 3 21 2 ]
```

CON? .


```
2:              "YES"
1: (-1.44444444444,
   .111111111111)
```

If the message "PARALLEL" appears, it means that the three lines are all parallel (and hence meet in a point at infinity).

Distance from a Point to a Line

$\boxed{10.1}$

This program, called D.PTL, takes a point (a, b) and a vector [p q r] from the stack and returns the distance from the point (a, b) to the line px + qy + r = 0.

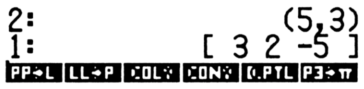
« → P L « P C→R 1 3 →ARRY L DOT ABS L ARRY→ DROP2 2 →ARRY ABS / » »

28S checksum: "78DA"
48SX checksum: # 22512d

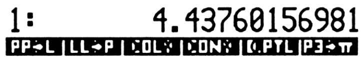
On the HP-48SX, the command ARRY→ must be replaced by OBJ→.

For example, to find the distance from the point (5, 3) to the line 3x + 2y - 5 = 0, type

(5 , 3 ENTER [3 , 2 , 5
CHS ENTER


```
2:              (5,3)
1:          [ 3 2 -5 ]
```

D.PTL .


```
1:      4.43760156981
```

Chapter 3. Functions and Limits

The capabilities of the HP graphing calculators make possible the investigation of functions in ways never seen in a calculator before. In this chapter, we first present ways to automate the graphing features of the calculators, and talk about plotting unusual and interesting functions. We mention limits, and present the bisection method for estimating zeros of functions.

An Enriched Graphing Environment

The HP-48SX has built into it automatic screen-saving, overdrawing, and range-finding. The following environment can be programmed into the HP-28S to give it the same capabilities.

Automatic Saving

On the HP-48SX, graphs are drawn onto a "canvas" (portion of memory) called PICT. This canvas can be viewed at any time by pressing $\boxed{\text{GRAPH}}$. For the HP-28S, we must save the canvas ourselves.

An environment for automatic screen-saving should contain, in addition to the containers PPAR and EQ, a container SCR for storing the screen and a routine RCLGR for recalling the last graph drawn. A sample environment would consist of the following, perhaps placed in a GRAPH subdirectory:

```
FDRA        RCLGR       SCR         EQ          PPAR        QUIT
```

Here is a description of each of these containers (in reverse order, so that as you create them they will appear on the menu in the desired order).

• QUIT contains the program « UP », and simply moves you up out of the subdirectory. Create it by typing « U P $\boxed{\text{ENTER}}$ ' Q U I T $\boxed{\text{STO}}$. This assumes that the program UP is on the top level of the USER menu.

• PPAR contains the plot parameters. It is most easily created by typing DEFPP $\boxed{\text{ENTER}}$, provided that you have the program DEFPP in the top level of the USER menu.

• EQ contains the current function whose graph is being drawn. It is used by the DRAW command in sketching the graph.

• SCR contains the binary string representation of the most recently-drawn screen.

• RCLGR is the program « SCR →LCD DGTIZ ». It simply recalls the contents of SCR and puts it onto the screen. The command DGTIZ puts the screen in interactive mode and activates the cursor keys.

• FDRA is the program

```
« STEQ CLLCD DRAW LCD→ 'SCR' STO DGTIZ ».
                                        28S checksum: "D106"
```

It takes an algebraic expression, an equation, or a program from level 1 of the stack, stores it in EQ, clears the screen, draws the graph, saves the screen in SCR, and activates the interactive mode on the screen.

To draw the graph, say of the function $f(x) = \sqrt{x^2 + 1}$, type the following: ' √ (X ^ 2 + 1 $\boxed{\text{ENTER}}$ $\boxed{\text{FDRA}}$. The graph will be in interactive mode. Press $\boxed{\text{ATTN}}$ to return to the normal stack display. Press $\boxed{\text{RCLGR}}$ to instantly recall the graph just drawn.

Overdrawing

On the HP-48SX, a graph is drawn on PICT on top of whatever is already there. If we do not want multiple graphs, we must erase the old one each time.

We can do a similar thing on the HP-28S. To draw the graph of a second function on top of a previous one, it is only necessary to recall the first graph before drawing the second one.

If it is desired to interact with a multiple graph, then the functions that created the graphs must also be kept for use again. This suggests that a list of the functions used should be maintained. That is exactly what the HP-48SX does; EQ becomes a list of functions.

We can therefore expand the above menu list by adding the following:

• EQS, a list of the functions used thus far in creating the latest picture.

• OVDRA (for OVerDRAw), the program

```
« DUP 1 →LIST EQS + 'EQS' STO STEQ SCR →LCD DRAW LCD→
'SCR' STO DGTIZ »,
                                        28S checksum: "61AD"
```

which takes a new function from the stack, adds it to the list of functions used thus far, recalls the previous picture, draws the new graph over it, and stores the new screen.

The program FDRA above should be replaced by the following:

• CLDRA (for CLear and DRAw), the program

```
« DUP 1 →LIST 'EQS' STO STEQ CLLCD DRAW LCD→ 'SCR' STO
DGTIZ »,
                                        28S checksum: "CE1"
```

which takes the function from the stack, puts it in a list all by itself in EQS, draws the graph, and saves the screen. Thus this program discards all previously-used functions, and starts over.

For example, to plot the graphs of f(x) = sin x and g(x) = cos x on the same screen, type ' S I N (X $\boxed{\text{ENTER}}$ $\boxed{\text{CLDRA}}$ $\boxed{\text{ON}}$ ' C O S (X $\boxed{\text{ENTER}}$ $\boxed{\text{OVDRA}}$.

Zooming

On the HP-48SX, zooming operations are built into the graphics environment, as described starting on p. 301 of the *HP-48SX Owner's Manual*. We can go a long way toward duplicating that environment on the HP-28S.

Automating the interactive features of the screen allows for easy zooming, in or out. The following routines can be added to the above menu, and used to modify the plot parameters and redraw the graphs.

- REDRA (for REDRAw), the program

```
« CLLCD EQS LIST→ 1 SWAP START STEQ DRAW NEXT LCD→
'SCR' STO DGTIZ »,
                                        28S checksum: "7C80"
```

which clears the screen, draws the graphs of each of the functions in the list EQS, and saves the picture.

- NWIN (for New WINdow), the program

```
« C→R → A B « C→R → C D « A C MIN B D MIN R→C PMIN
A C MAX B D MAX R→C PMAX REDRA » » »,
                                        28S checksum: "DF41"
```

which takes two points, a pair of diagonally opposite corners of the new window desired, adjusts the plot parameters to create the new window, and redraws the previous graph or graphs. The diagonally opposite points are usually supplied by digitizing from the graph, and the choice of which pair or in which order they are entered does not matter.

- ZOUT (for Zoom OUT), the program

```
    « 3 ROLL CENTR *H *W REDRA »,
                                        28S checksum: "380B"
```

which adjusts the plot parameters so that a specified point shows at the center of the screen and distances are multiplied by factors m_x in the horizontal direction and m_y in the vertical direction, and then redraws the graphs. The center is taken from level 3, m_x from level 2, and m_y from level 1. The center is usually specified by digitizing from the picture. If m_x and m_y are greater than 1, then the picture is compressed to fit more of it into the window.

- ZIN (for Zoom IN), the program

```
    « INV SWAP INV SWAP ZOUT »,
                                        28S checksum: "BB2C"
```

which does the same thing that ZOUT does, except that distances are divided by the factors m_x and m_y instead of multiplied. If m_x and m_y are greater than 1, then the picture is magnified but less of it is visible.

Automatic Range-finding

One of the first problems you will encounter is that when an unfamiliar graph is plotted, you have no idea what the most appropriate plot parameters are, and the graph is often not visible. One possible approach to an automatic solution to this problem is to have the calculator sample the values of the function and automatically adjust the window size to accommodate the graph. This requires telling the machine the interval over which the graph is to appear.

On the HP-48SX, the command $\boxed{\text{AUTO}}$ on the PLOT PLOTR menu does just that, using the x-range specified. The same thing is built into zooming options on the GRAPH menu; see pp. 302 "Z-BOX" and 305 "XAUTO" of the *HP-48SX Owner's Manual.*

For the HP-28S, the following program, called AUDRA (for AUto-DRAw), takes the function from level 3 and the x-range [A, B] from levels 2 and 1, and tries to construct an "appropriate" y-range for the graph.

```
« →NUM SWAP →NUM DUP2 IF < THEN SWAP END DUP2 - 10 /
→ B A H « STEQ "M" CLΣ A 'X' STO 1 11 START IFERR EQ
→NUM DUP IF IM 0 ≠ THEN DROP ELSE 1 →ARRY Σ+ END THEN
END H 'X' STO+ NEXT A MINΣ 1.1 * R→C PMIN B MAXΣ 1.1 *
R→C PMAX CLΣ 'X' PURGE » WHILE "M" ≠ REPEAT END EQ
CLDRA »
                                        28S checksum: "83D1"
```

For example, to plot the graph of f(x) = x sin 2x over the interval [-3, 3], type ' X * S I N (2 * X $\boxed{\text{ENTER}}$ 3 $\boxed{\text{CHS}}$ $\boxed{\text{ENTER}}$ 3 $\boxed{\text{ENTER}}$ $\boxed{\text{AUDRA}}$.

Plotting Unusual Functions

Occasionally functions occur whose formulas cannot be specified in purely algebraic terms. Such functions include the greatest integer function, the absolute value function, and perhaps some others. Idiosyncrasies of the calculator make it difficult to plot graphs of some other functions, such as those involving fractional exponents. Yet other functions are defined by different formulas for different parts of their domains. Here we look at some ways to work around the difficulties imposed by these types of functions.

ABS and FLOOR

$\boxed{1.4}$

The absolute value function, f(x) = |x|, is known to the calculator as ABS(X). ABS is found on the REAL menu on the HP-28S and on the MTH PARTS menu on the HP-48SX. For example, to enter the function f(x) = | 2x - 5 |, type 'ABS(2*X-5)'.

The greatest integer function, f(x) = $\lfloor x \rfloor$, is known to the calculator as the FLOOR function, found on the same menus as ABS. For example, to enter the function f(x) = 3$\lfloor x - 1 \rfloor$, type '3*FLOOR(X-1)'.

Fractional Exponents

If the calculator raises a negative number to a fractional power, it automatically puts the result in complex number form, even when the result is a real number. That means that portions of a graph will not be shown when the fractional power of a negative number is involved.

To get the machine to show the entire graph, we use the ABS function in conjunction with the SIGN function, found on the same menu. The SIGN function is defined by

$$\mathrm{SIGN}(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x = 0 \\ -1 & \text{if } x < 0 \end{cases}.$$

For example, to enter the function $f(x) = (x - 2)^{1/3}$, we write the product

'SIGN(X-2)*(ABS(X-2))^(1/3)'.

This works, because no negative numbers appear to fractional powers, and the cube root of a negative number is the negative of the cube root of its absolute value.

Routines as Functions

Sometimes functions are given in "installments," or are defined by different formulas for different parts of their domains. For example, the function

$$f(x) = \begin{cases} \sin x & \text{if } x < 0 \\ 1 - \frac{1}{4}x^2 & \text{if } x \geq 0 \end{cases}$$

can be typed as "IFTE(X<0,SIN(X),1-X^2/4)'. It can also be graphed by any program that uses DRAW by using the procedure

```
« IF X 0 < THEN 'SIN(X)' ELSE '1-X^2/4' END »
```

as the function. As another example, the function

$$f(x) = \begin{cases} 2x & \text{if } x \leq -1 \\ x^2 - 1 & \text{if } -1 < x \leq 2 \\ 1 - x & \text{if } x > 2 \end{cases}$$

can be represented by the procedure

```
« IF X -1 ≤ THEN '2*X' ELSE IF -1 X < X 2 ≤ AND THEN 'X^2-
1' ELSE '1-X' END END ».
```

Limits

$\boxed{2.1\text{-}2.4}$

The evaluation of limits via calculator is sometimes possible, in perhaps a couple of ways. One is graphical, and the other is computational, but these are really the same thing when you think about it.

Graphical Evaluation

To examine the limit $\lim_{x \to c} f(x)$, try graphing the function f over an interval containing c. You may then zoom in toward c to "blow up" the picture, and arrive at a conclusion about the limit in that way.

Computational Evaluation

Another way to examine the limit $\lim_{x \to c} f(x)$ is by using the Solver with f(x) as the expression, and evaluate at values successively nearer to c. You may be able to arrive at a conclusion about the limit in that way.

Bisection Method

$\boxed{2.5}$

The bisection method is a means of systematically closing in on a zero of a function when an interval containing the zero is known. The method uses the continuity of the function and the fact that the function has opposite signs at the endpoints of the interval to locate the zero. Successively more precise locations are found by repeatedly bisecting the interval and retaining the half that contains the zero.

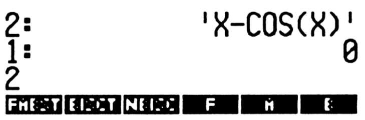
The following programs enable one to find an arbitrarily short interval containing a zero of a given function f, once an interval [a, b] is found for which f(a) and f(b) have opposite signs. The first program is just a simple routine that stores the function and the initial interval; it has the great value, however, of reminding you how to set up to use the bisection method. The program is

```
« 'B' STO 'A' STO 'F' STO A B 2 →ARRY »,
                                        28S checksum: "702F"
                                        48SX checksum: # 22501d
```

and is stored under the label FABST. The name of the program reminds you to enter the function f and the endpoints a and b of the interval, in that order, and then press $\boxed{\text{FABST}}$. The function and the interval are stored for use by the next program, and the initial interval is placed on the stack for reference.

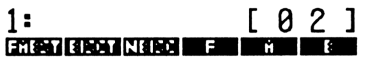
For example, the set up the bisection method for the function f(x) = x - cos x on the interval [0, 2], type

'X - C O S (X $\boxed{\text{ENTER}}$ 0 $\boxed{\text{ENTER}}$ 2



```
2:          'X-COS(X)'
1:                   0
2
```

$\boxed{\text{FABST}}$.



```
1:           [ 0 2 ]
```

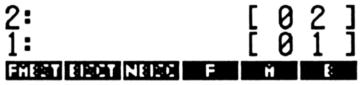
The second program actually does the computation. It finds the midpoint of the interval, tests to see in which half the zero lies, and resets the appropriate endpoint so as to retain the correct half of the original interval. It also checks to see that the initial interval really does contain a zero, and if we happen to hit the zero exactly, that is also made known. The new interval is placed on the stack. Here is the program, which I call BISCT.

```
« A B + 2 / →NUM 'C' STO A 'X' STO F →NUM B 'X' STO F
→NUM * IF 0 > THEN "SAME SIGN" ELSE A 'X' STO F →NUM C
'X' STO F →NUM * IF DUP 0 == THEN DROP C "IS A ZERO"
ELSE IF 0 < THEN C 'B' STO ELSE C 'A' STO END A B 2
→ARRY END END { C X } PURGE »
```

28S checksum: "FEA5"
48SX checksum: # 46422d

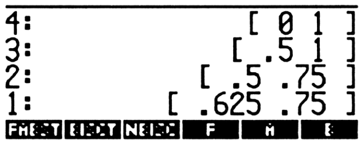
For example, once the function and initial interval are stored, press

$\boxed{\text{BISCT}}$ to get the next interval.



```
2:           [ 0 2 ]
1:           [ 0 1 ]
```

Keep pressing $\boxed{\text{BISCT}}$ for successive intervals.



```
4:           [ 0 1 ]
3:          [ .5 1 ]
2:        [ .5 .75 ]
1:      [ .625 .75 ]
```

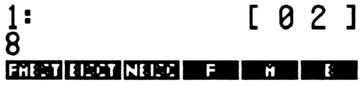
You can also automate the application of the bisection method a specified number of times. Here is a little program that essentially just presses the $\boxed{\text{BISCT}}$ button n times. It is called NBISC.

```
« 1 SWAP START BISCT NEXT »
```

28S checksum: "D823"
48SX checksum: # 59739d

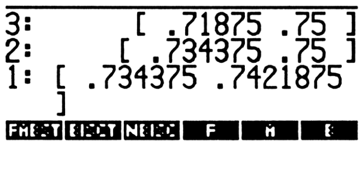
This program takes the number n from the stack, and calls BISCT that many times. For example, having stored f, a, and b already, if you want to press $\boxed{\text{BISCT}}$ eight times the easy way, type

8



```
1:           [ 0 2 ]
8
```

$\boxed{\text{NBISC}}$.



```
3:      [ .71875 .75 ]
2:     [ .734375 .75 ]
1: [ .734375 .7421875
   ]
```

It is convenient to organize these programs into their own subdirectory, perhaps named BISEC. The complete menu under BISEC might be

```
FABST   BISCT   NBISC   F   A   B
```

You might also add the program « UP » (« UPDIR » for the HP-48SX) to this menu, under the label QUIT.

Chapter 4. Derivatives and Their Applications

The HP symbol-manipulating calculators can find derivatives of all the elementary functions. Derivatives are presented in unsimplified form, illustrating the various differentiation formulas. Specifying the independent variable is necessary, so that partial differentiation is also possible. That makes implicit differentiation possible, for which a simple program is presented.

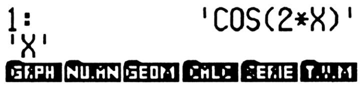
Finding Derivatives

$\boxed{3.3\text{-}3.5}$

To find the derivative of a function, enter the function, specify its independent variable, and press $\boxed{\text{d/dx}}$ on the HP-28S or $\boxed{\partial}$ on the HP-48SX. The derivative is displayed.

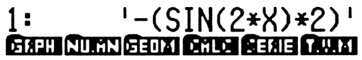
For example, to find the derivative of f(x) = cos 2x, type

' C O S (2 * X $\boxed{\text{ENTER}}$ ' X



```
1:            'COS(2*X)'
'X'
```

$\boxed{\text{d/dx}}$.

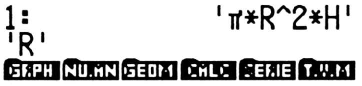


```
1:       '-(SIN(2*X)*2)'
```

The derivative is presented in the form that illustrates the chain rule. You can also see why being in radian mode is important, if you happened to be in degree mode.

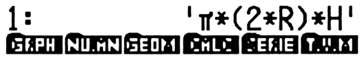
As another example, suppose that from the formula V = $\pi r^2 h$ you wish to find $\frac{dV}{dr}$. Enter the right-hand side of the formula and differentiate, by typing

' π * R ^ 2 * H $\boxed{\text{ENTER}}$ ' R



```
1:             'π*R^2*H'
'R'
```

$\boxed{\text{d/dx}}$.



```
1:           'π*(2*R)*H'
```

Simplifying

Since derivatives are given in unsimplified form, you can simplify them somewhat by using $\boxed{\text{COLCT}}$ and $\boxed{\text{EXPAN}}$ on the ALGEBRA menu. For many applications, however, a simplified form is not needed, and the calculator has no trouble working with the unsimplified form. It sometimes takes longer, however, in unsimplified form.

Evaluating

Evaluating a derivative is done in exactly the same ways that other expressions are evaluated. There is one additional means, however, that you should be aware of.

If a function f uses the variable X, and a container named X (containing the number c) is on the current level of the USER or VAR menu, then when the $\boxed{\text{d/dx}}$ command is given, not only is the derivative computed, but it is evaluated at the number c contained in X. The value f '(c), not the derivative f ', is returned. This can be convenient when you want to evaluate the derivative, but it is not so convenient if you really want the function f '. If you want the function f ', make sure that no container named X is on the USER or VAR menu.

Higher-order Derivatives

$\boxed{3.6}$

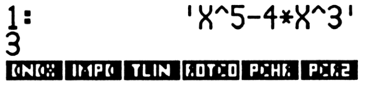
Finding higher-order derivatives is no different on the calculator than it is by hand; we just differentiate again. We must specify the independent variable each time, though.

If you want to automate the finding of higher-order derivatives, you can construct a program that will return the particular higher-order derivative that you want. Here is one that will take the function from level 2 and the order n of the derivative from level 1 and return the nth derivative; this program is called DNDX.

```
« 1 SWAP START 'X' ∂ COLCT NEXT »
                              28S checksum: "DD6A"
                              48SX checksum: # 32234d
```

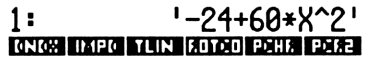
For example, if you want the 3rd derivative of $f(x) = x^5 - 4x^3$, type

' X ^ 5 - 4 * X ^ 3 $\boxed{\text{ENTER}}$ 3



```
1:            'X^5-4*X^3'
3
```

$\boxed{\text{DNDX}}$.



```
1:           '-24+60*X^2'
```

Note that the programs assumes that X is the variable; the name of the program reminds you of that.

Implicit Differentiation

$\boxed{3.8}$

The HP calculators can be programmed to produce the derivative of an implicit function. The theory behind it is couched in partial derivatives; the formula used is that if f(x, y) = 0, then $\frac{dy}{dx} = -\frac{\partial f/\partial y}{\partial f/\partial x}$. Here is the program, called IMPD, which starts with the expression f(x, y) on the stack and returns both the expression f and the derivative $\frac{dy}{dx}$.

```
« DUP 'Y' ∂ COLCT 'FY' STO DUP 'X' ∂ NEG COLCT FY / COLCT
'FY' PURGE »
                              28S checksum: "E321"
                              48SX checksum: # 6634d
```

For example, given $x^2 + y^3 = 5$, the derivative is found implicitly by typing

' X ^ 2 + Y ^ 3 - 5 $\boxed{\text{ENTER}}$

```
1:            'X^2+Y^3-5'
 CNCX  IMPD  TLIN  ROTCO  PCHR  PCR2
```

$\boxed{\text{IMPD}}$.

```
2:            'X^2+Y^3-5'
1: '-(.666666666666*X*
   Y^-2)'
 CNCX  IMPD  TLIN  ROTCO  PCHR  PCR2
```

Try pressing $\boxed{\to Q}$ to make the result look better.

```
2:            'X^2+Y^3-5'
1:       '-(2/3*X*Y^-2)'
 CNCX  IMPD  TLIN  ROTCO  PCHR  PCR2
```

Tangent Lines

$\boxed{3.1\text{ff}}$

The simplest application of the derivative of a function is to find the tangent line to the graph of the function at a specified point. For example, if y = f(x) is given, then the tangent line to the graph at (c, f(c)) is y - f(c) = f '(c) (x - c). The derivative f '(c) is easily found, as described above, and a program can be written to take the function f and the number c from the stack and produce the equation of the tangent line. It might look something like this, which I call TLIN:

```
« 'X' STO DUP 'X' ∂ 'X' X - * SWAP EVAL 'Y' SWAP - SWAP =
'X' PURGE »
```

28S checksum: "2370"
48SX checksum: # 60926d

For example, to find the line tangent to the curve $y = x^2 - 4$ at the point (3, 5), type

' X ^ 2 - 4 $\boxed{\text{ENTER}}$ 3

```
1:                'X^2-4'
3
 CNCX  IMPD  TLIN  ROTCO  PCHR  PCR2
```

$\boxed{\text{TLIN}}$.

```
1:         'Y-5=6*(X-3)'
 CNCX  IMPD  TLIN  ROTCO  PCHR  PCR2
```

This program can be modified to produce the equation of the normal line, just by inserting the commands INV NEG after the ∂ symbol, for only the slope is different.

In the graphics environment that allows overdrawing, the tangent line to a curve at a point can be specified by digitizing the point of tangency, and the line can be drawn on the screen with the curve. The following program for the HP-28S, called TANL, does the job.

```
« C→R DROP DUP 'X' STO 'X' SWAP - EQ 'X' ∂ * EQ EVAL +
OVDRA 'X' PURGE »
```

28S checksum: "5CE1"

For the HP-48SX, the program is

```
« 1 'N' STO EQ DUP IF TYPE 5 == THEN OBJ→ DUP 'N' STO
PICK 'CEQ' STO N DROPN ELSE 'CEQ' STO EQ 1 →LIST STEQ END
EQ SWAP C→R DROP DUP 'X' STO 'X' SWAP - CEQ 'X' ∂ * CEQ
EVAL + STEQ DRAW 'X' PURGE EQ 1 →LIST + STEQ { CEQ X }
PURGE »
                                        48SX checksum: # 9958d
```

For example, suppose you have just drawn the graph of a function f, and while the graph is still up you position the cursor at the desired point of tangency (only the x-coordinate of the point is used by the program). Press INS or ENTER to digitize the point, then ON, and then press TANL. The process can be repeated when the new picture is displayed. This program can also be modified to produce the normal line.

Newton's Method

4.2

Another application of differentiation is Newton's method for estimating a zero of a function. The method starts with the function f and an estimate x_1 of the zero, and uses the iteration formula $x_2 = \frac{f(x_1)}{f'(x_1)}$ to produce a (hopefully) better estimate.

A simple environment for using Newton's method on the HP calculators consists of two programs, FSTO for storing the function f and its derivative, and NEWT for computing x_2, given x_1. Here is FSTO:

```
« DUP 'F' STO 'X' ∂ 'FPR' STO »
                                        28S checksum: "2BDC"
                                        48SX checksum: # 13860d
```

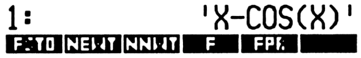
Here is NEWT:

```
« 'X' STO X X F EVAL FPR EVAL / - 'X' PURGE »
                                        28S checksum: "DA17"
                                        48SX checksum: # 50627d
```

The label FSTO reminds you to enter the expression for f(x); pressing FSTO not only stores the function f, but also computes the derived function f ' and stores it under FPR. Then, knowing that Newton's method requires an initial guess, you enter a number x_1, close to the zero you wish to find, and press NEWT. The new estimate x_2 appears, along with x_1, so that you can compare them. If you wish another estimate, just press NEWT again, and another estimate appears. Continue pressing NEWT until you have the accuracy you need, or until the new estimate does not differ from the previous one.

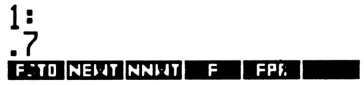
For example, to find the zero of f(x) = x - cos x that is near .7, type

'X - C O S (X ENTER



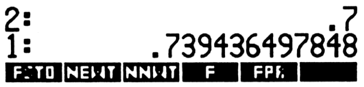
```
1:               'X-COS(X)'
FSTO NEWT NNWT  F   FPR
```

FSTO .7



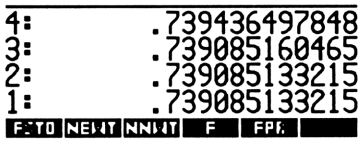
```
1:
.7
```

NEWT .



```
2:                .7
1:     .739436497848
```

Press NEWT three or four more times.



```
4:     .739436497848
3:     .739085160465
2:     .739085133215
1:     .739085133215
```

You can also automate the pressing of NEWT a specified number of times. The following program, called NNWT, takes x_1 from level 2 and a number n from level 1, and presses NEWT n times for you. It assumes that F and FPR are already stored.

```
« 1 SWAP START NEWT NEXT »
```

28S checksum: "5D99"
48SX checksum: # 1660d

It is convenient to put these programs into their own subdirectory, perhaps called NWTN. The complete menu might be

FSTO NEWT NNWT F FPR QUIT

Extrema

4.4,4.5

Finding an extremum of the function f usually involves solving an equation of the form f '(x) = 0. Such an equation can be solved in a variety of ways on the HP calculators, as we discussed in Chapter 2. The calculator will even do the differentiation for you, as discussed above.

On the HP-48SX, finding an extremum in a graphics setting is built in. After a graph is plotted, position the cursor near the extremum and press EXTR on the GRAPH FCN menu. On the HP-28S, an extremum can be found from a graph by digitizing an approximate extremum and then using the following program, EXTR, to find the extremum precisely.

```
« C→R DROP EQ 'X' PURGE 'X' ∂ 'X' 3 ROLL ROOT DUP 'X'
STO EQ EVAL R→C 'X' PURGE »
```

28S checksum: "AA1C"

This program takes a point from the stack and returns the nearest point (c, f(c)) for which f '(c) = 0.

Points of Inflection

$\boxed{4.3}$

A point of inflection of the function f is usually found by solving an equation of the form f"(x) = 0. In a graphics setting, an approximate inflection point can be digitized, and then the following program, INFL, will return the inflection point precisely.

```
« C→R DROP EQ 'X' PURGE 'X' ∂ 'X' ∂ 'X' 3 ROLL ROOT DUP
'X' STO EQ EVAL R→C 'X' PURGE »
                                   28S checksum: "C4E6"
                                   48SX checksum: # 26417d
```

This program takes a point from the stack and returns the nearest point (c, f(c)) for which f "(c) = 0. This program can be followed by TANL, for example, to draw an inflectional tangent to a graph.

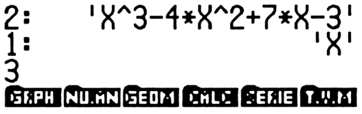
Antiderivatives

$\boxed{4.6}$

The HP symbol-manipulating calculators have a limited ability to find antiderivatives of functions. The HP-28S can find antiderivatives (perform symbolic integration) only for polynomial functions. To do so, the stack must have the polynomial integrand in level 3, the variable of integration in level 2, and the degree of the integrand in level 1. Then $\boxed{\int}$ will return the antiderivative of the polynomial, except for the constant of integration, which you must supply.

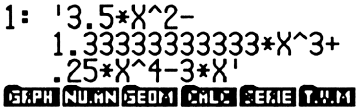
For example, to find the antiderivative of f(x) = $x^3 - 4x^2 + 7x - 3$, type

'X ^ 3 - 4 * X ^ 2 + 7 * X - 3 $\boxed{\text{ENTER}}$ ' X $\boxed{\text{ENTER}}$ 3



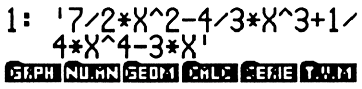
```
2:      'X^3-4*X^2+7*X-3'
1:                   'X'
3
```

$\boxed{\int}$.



```
1: '3.5*X^2-
   1.33333333333*X^3+
   .25*X^4-3*X'
```

It is appropriate to apply $\boxed{\to Q}$ to the result.



```
1: '7/2*X^2-4/3*X^3+1/
   4*X^4-3*X'
```

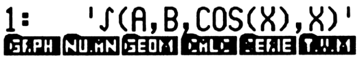
For functions other than polynomials, the HP-28S will do "approximate" symbolic integration, as discussed in Chapter 8.

On the HP-48SX, antidifferentiation is done in the context of definite integration, with variable limits. The procedure is to enter the integral and then evaluate it.

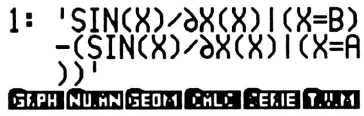
There are three ways to enter the integral. The most elegant way is to use the EquationWriter, as illustrated on pp. 429-430 of the *HP-48SX Owner's Manual.*

For example, to enter $\int_a^b \cos(x)\,dx$ in the EquationWriter, type

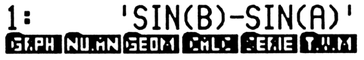
$\boxed{\text{EQUATION}}$ $\boxed{\int}$ A $\boxed{\blacktriangleright}$ B $\boxed{\blacktriangleright}$ $\boxed{\text{COS}}$ X $\boxed{\blacktriangleright}$ $\boxed{\blacktriangleright}$ X $\boxed{\text{ENTER}}$. The integral is translated into algebraic form on the stack.


```
1:    '∫(A,B,COS(X),X)'
```

To evaluate, press $\boxed{\text{EVAL}}$,

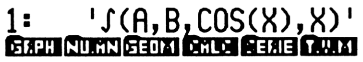

```
1: 'SIN(X)/∂X(X)|(X=B)
   -(SIN(X)/∂X(X)|(X=A
   ))'
```

and then $\boxed{\text{EVAL}}$ again.


```
1:       'SIN(B)-SIN(A)'
```

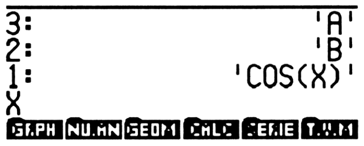
A second way to enter the integral is to type it into the command line in algebraic form directly. To enter the above integral again, type

' $\boxed{\int}$ A , B , $\boxed{\text{COS}}$ X $\boxed{\blacktriangleright}$, X $\boxed{\text{ENTER}}$.

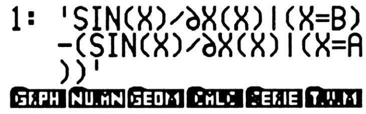

```
1:    '∫(A,B,COS(X),X)'
```

The third way is to put the limits, integrand, and variable of integration on the stack, followed by the $\boxed{\int}$ command. To evaluate the same integral in this way, type

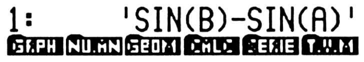
A $\boxed{\text{ENTER}}$ B $\boxed{\text{ENTER}}$ ' $\boxed{\text{COS}}$ X $\boxed{\text{ENTER}}$ X


```
3:                  'A'
2:                  'B'
1:             'COS(X)'
X
```

$\boxed{\int}$


```
1: 'SIN(X)/∂X(X)|(X=B)
   -(SIN(X)/∂X(X)|(X=A
   ))'
```

$\boxed{\text{EVAL}}$.


```
1:       'SIN(B)-SIN(A)'
```

The HP-48SX can give the antiderivatives of polynomials and of any function that is the derivative of a built-in function, as described on p. 429 of the *HP-48SX Owner's Manual.* You can experiment to find out what the HP-48SX will and won't do in symbolic integration.

Chapter 5. Definite Integrals and Their Applications

$\boxed{5.2\text{ff}}$

The HP calculators do a very fine job of evaluating definite integrals. The *HP-28S Reference Manual* discusses numerical integration, starting on p. 101, and the *HP-48SX Owner's Manual*, on p. 432. You can also program your own numerical integration routines, as we will discuss in Chapter 7.

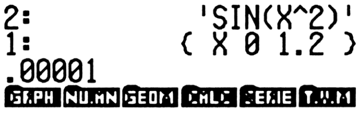
The Built-in Integrator

In both calculators, numerical integration requires input of the integrand, the limits, the variable of integration, and an accuracy factor, and returns the approximate value of the integral and an uncertainty number (which is almost certainly greater than the difference between the given value and the actual value). However, the two calculators do these things in quite different ways.

Suppose we wish to evaluate the definite integral $\int_a^b f(x)\,dx$. On the HP-28S, put the integrand f(x) on level 3 in algebraic form with X as the variable, put the list { X a b } on level 2, and put accuracy factor on level 1. (It is usually sufficient to use something like .00001 as the accuracy factor.) Then press $\boxed{\int}$, and the calculator returns the approximate value of the integral to level 2 and the uncertainty number to level 1.

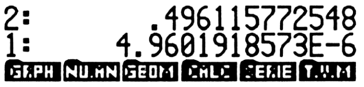
For example, to evaluate $\int_0^{1.2} \sin(x^2)\,dx$, type

' S I N (X ^ 2 $\boxed{\text{ENTER}}$ { X , 0 , 1.2 $\boxed{\text{ENTER}}$ .00001



```
2:          'SIN(X^2)'
1:        { X 0 1.2 }
.00001
```

$\boxed{\int}$.



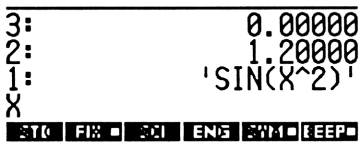
```
2:      .496115772548
1:    4.9601918573E-6
```

Read the value from level 2, and the uncertainty number from level 1.

For the HP-48SX, the integral is entered as described above, in the section on Antiderivatives. Probably the smallest number of keystrokes is required if the stack is used: put a in level 4, b in level 3, f(x) in level 2, x in level 1, and press $\boxed{\int}$ and then $\boxed{\to\text{NUM}}$ to get the approximate value of the integral. The uncertainty number is stored in a container named IERR on the VAR menu. The accuracy factor is specified by setting the FIX mode to the desired degree of precision. If you want 10^{-n} as the accuracy factor, enter n and press $\boxed{\text{FIX}}$ on the MODES menu.

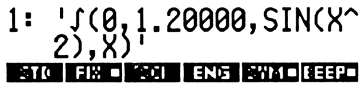
For example, to evaluate the integral above with the same accuracy, type

$\boxed{\text{MODES}}$ 5 $\boxed{\text{FIX}}$ 0 $\boxed{\text{ENTER}}$ 1.2 $\boxed{\text{ENTER}}$
' $\boxed{\text{SIN}}$ X $\boxed{y^x}$ 2 $\boxed{\text{ENTER}}$ X



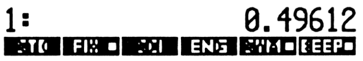
```
3:            0.00000
2:            1.20000
1:          'SIN(X^2)'
X
```

$\boxed{\int}$



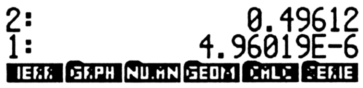
```
1: '∫(0,1.20000,SIN(X^
   2),X)'
```

$\boxed{\to\text{NUM}}$.



```
1:            0.49612
```

Press $\boxed{\text{IERR}}$ on the VAR menu to see the uncertainty number.



```
2:            0.49612
1:         4.96019E-6
```

Areas, Volumes, and Arc Lengths

$\boxed{5.6,8.1\text{-}8.3}$

The value of a definite integral can be interpreted as area, volume, arc length, or something else, depending on how the definite integral is set up. In every case, the calculator can evaluate the definite integral in the same way. It is in setting up the integral that the differences occur, and even there the calculator can be of help. The following examples will give you the idea.

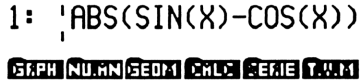
The area between curves y = f(x) and y = g(x) over the interval [a, b] is

$$\int_a^b | f(x) - g(x) | \, dx \quad .$$

To set up to evaluate this integral numerically, enter the integrand by typing f(x), then g(x); then subtract, and then take the absolute value by pressing $\boxed{\text{ABS}}$ or by typing A B S $\boxed{\text{ENTER}}$.

For example, the integrand of $\int_{-\pi}^{\pi/5} | \sin x - \cos x | \, dx$ is created by typing

' S I N (X $\boxed{\text{ENTER}}$ ' C O S (X $\boxed{\text{ENTER}}$
$\boxed{-}$ A B S $\boxed{\text{ENTER}}$.

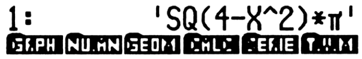


```
1: 'ABS(SIN(X)-COS(X))
```

If the function f is positive over [a, b] and the region under f is rotated about the x-axis to generate a solid of revolution, the volume of the solid is $\int_a^b \pi[f(x)]^2 \, dx$. To enter the integrand, just type in f, square it, and multiply by π.

For example, to find the volume of the solid generated by revolving the region under f(x) = 4 - x^2 over [0, 2] about the x-axis, we must evaluate the integral $\int_0^2 \pi(4 - x^2)^2\,dx$. We can form the integrand by typing

' 4 - X ^ 2 $\boxed{\text{ENTER}}$ $\boxed{x^2}$ π * .

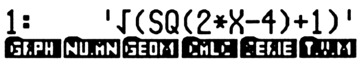


```
1:          'SQ(4-X^2)*π'
```

The length of the graph of y = f(x) over [a, b] is $\int_a^b \sqrt{1 + [f'(x)]^2}\,dx$. To set up this integrand, enter f, differentiate it, square it, add 1, and take the square root.

For example, to find the length of the curve y = x^2 - 4x + 5 over [-1, 3], we must evaluate the integral $\int_{-1}^{3} \sqrt{1 + [\frac{d}{dx}(x^2 - 4x + 5)]^2}\,dx$. The integrand can be created by typing

' X ^ 2 - 4 * X + 5 $\boxed{\text{ENTER}}$ ' X $\boxed{\text{ENTER}}$
$\boxed{d/dx}$ $\boxed{x^2}$ 1 + $\boxed{\sqrt{\ }}$.



```
1:     '√(SQ(2*X-4)+1)'
```

Chapter 6. Transcendental Functions

All of the elementary functions are built into the HP calculators, and evaluating a function is just the push of a button.

LN, e, and EXP

$\boxed{\boxed{6.1\text{-}6.4}}$

The natural logarithm function is LN (on the LOGS menu of the HP-28S). To find the natural logarithm of x, enter x and press $\boxed{\text{LN}}$. If x is not a positive number, then LN returns a complex number result.

The number e is the base of the natural logarithm, and satisfies the property ln e = 1. The calculator knows e as a symbolic constant; enter a lower-case e, and the calculator puts 'e' on the stack. Press $\boxed{\to\text{NUM}}$ to see the value of e.

The natural exponential function is $f(x) = e^x = \exp(x)$. To evaluate e^x, enter x and press $\boxed{\text{EXP}}$ (on the LOGS menu of the HP-28S) or $\boxed{e^x}$. The key sequence 1 $\boxed{\text{EXP}}$ or 1 $\boxed{e^x}$ also gives the value of the number e.

Other functions on the LOGS menu are discussed in the *HP-28S Reference Manual*, beginning on p. 133. The same functions are found on the MTH HYP menu of the HP-48SX, described on p. 137 of the *HP-48SX Owner's Manual.*

The derivatives of the LN and EXP functions are built into the calculator, and can be used in differentiation and integration problems just as other functions are.

It may be that a number on the stack is the logarithm or exponential of a rational number. The program « LN →Q EXP », which is called →EXQ, will transform the decimal into the exponential of a rational number. The program « EXP →Q LN », which is called →LNQ, will transform the decimal into the logarithm of a rational number. Both of these programs give spurious results if the number on the stack is not actually what you thought it was.

Inverse Trigonometric Functions

$\boxed{\boxed{6.6}}$

The inverse trigonometric functions were discussed in Chapter 2, and the derivatives of ASIN, ACOS, and ATAN are known to the calculator. The derivatives of the other inverse trig functions can be computed from these, using the formulas given in Chapter 2.

Hyperbolic Functions

6.7

The hyperbolic functions SINH, COSH, and TANH are given on the LOGS or MTH HYP menu, along with their inverses. The other hyperbolic functions are coth x = $\frac{1}{\tanh x}$, sech x = $\frac{1}{\cosh x}$, and csch x = $\frac{1}{\sinh x}$. Their values are found as for other functions; just enter x and press the appropriate button, and then use the $\boxed{1/x}$ button for the latter three.

Applications

6.4

Many applications of the transcendental functions are based on formulas that lend themselves well to treatment by the Solver. For example, the exponential growth model,

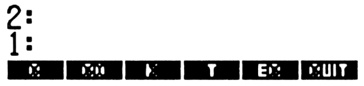
$$Q = Q_0e^{kt},$$

can be typed as 'Q=Q0*EXP(K*T)'. Store this as the equation by entering the SOLV menu and pressing $\boxed{\text{STEQ}}$, and then press $\boxed{\text{SOLVR}}$. The values of the known variables for a given problem can be stored easily and the remaining variable solved for.

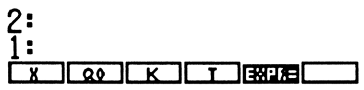
When you return to the USER or VAR menu, variables for the unknowns that were used will be present; you will have to purge them if you don't want them there. A solution to the problem of having to purge variables every time you use the Solver is to create a subdirectory into which you can put the formula to be used, together with containers for the variables in the formula. Then when you get through using the Solver, just exit the subdirectory and leave the variables there. For example, the above equation might be placed in a subdirectory called EXGR (for "EXponential GRowth model"), the complete menu of which might look like

Q   Q0   K   T   EQ   QUIT

Store the equation 'Q=Q0*EXP(K*T)' in EQ, and store zeros in the other containers, just to get them on the menu. QUIT would be the program « UP ». Then, whenever you want to work with the exponential growth model, enter the subdirectory EXGR,



and then press $\boxed{\text{SOLVR}}$ on the SOLV menu.



The equation is automatically set up for you, because it is already stored in EQ. Similar subdirectories could be set up for other applications, such as the exponential decay model, the logistic model, and so forth.

Chapter 7. Numerical Integration Theory

$\boxed{7.8}$

In first-year calculus, you learn to approximate definite integrals using the trapezoidal rule or Simpson's rule, which are simple approximation techniques that work very well for most definite integrals. Here we will present a few simple programs that develop these rules, as well as the rectangle rules (Riemann sum rules).

Begin by creating a subdirectory for numerical integration. We will call it N.INT for Numerical INTegration. Type

'N.INT $\boxed{\text{ENTER}}$

```
1:                 'N.INT'
GRPH NU.AN GEOM CALC SERIE T.V.M
```

C R D I R $\boxed{\text{ENTER}}$.

```
1:
N.INT GRPH NU.AN GEOM CALC SERIE
```

Then press $\boxed{\text{N.INT}}$ to enter the new subdirectory.

```
1:
```

On the HP-28S, create an easy exit by storing the program « UP » under the label QUIT (see p. 6).

```
1:
QUIT
```

Our aim is to approximate the integral $\int_a^b f(x)\,dx$. To do so, we will have to give the calculator the function f and the limits a and b. We will create a little program that makes this easy to do. Type in the following program:

```
« 'B' STO 'A' STO 'F' STO »
```

28S checksum: "E593"
48SX checksum: # 59581d

We will store this program under the label FABST, reminding us to put the entries on the stack in the order f, then a, then b. Once you have entered the above program, typing 'FABST $\boxed{\text{STO}}$ will store it for us.

To test our little storage program, let's prepare to evaluate $\int_1^2 \frac{1}{x}\,dx$. Type the following:

' 1 ÷ X $\boxed{\text{ENTER}}$ 1 $\boxed{\text{ENTER}}$ 2

```
2:                   '1/X'
1:                       1
2
FABST QUIT
```

$\boxed{\text{FABST}}$.

```
1:
  F    A    B  FABST QUIT
```

You will see three new containers in the menu, F, A, and B. You can press them to recall their contents if you wish to see whether everything worked properly.

Riemann Sum Rules

$\boxed{7.8}$

The first approximation we want to consider is a Riemann sum rule. This just uses the definition of the definite integral, which is the limit of Riemann sums, to get an approximation as follows. First, the interval [a, b] is subdivided into a number n of subintervals, all of the same length $h = \frac{b - a}{n}$. Next, in each subinterval a value t is chosen, and a rectangle of height f(t) and width h is created. The area of the rectangle is approximately the area under the function f. Finally, the sum of all such areas of rectangles is formed, and that is approximately the value of the integral.

The next step, then, is to tell the calculator now many subintervals we want. The following program stores the number n, and also calculates the number h and stores it, using the previously stored values of a and b.

```
      « 'N' STO B A - N / →NUM 'H' STO »
                                          28S checksum: "433A"
                                          48SX checksum: # 29449d
```

We will store this program under the label NSTO, reminding us that it stores n. To test it out, choose the number 4 for n, typing

4 $\boxed{\text{NSTO}}$.

You should see new containers for N and H.

The most commonly used Riemann sum rules are the left endpoint rule, the right endpoint rule, and the midpoint rule. In each of these, the numbers t from the subintervals are always spaced a distance h apart. The next program forms the basis of each of these Riemann sum rules, and creates the sum that each of them uses. We will call this program SUM:

```
« 'X' STO 0 1 N START F →NUM + H 'X' STO+ NEXT H * 'X' PURGE »
                                          28S checksum: "FAA1"
                                          48SX checksum: # 3683d
```

The command STO+ is found on the STORE menu on the HP-28S and on the *right-shifted* MEMORY menu on the HP-48SX.

This program takes a number from the stack as the starting point, stores it as X, and uses it to evaluate the function f. Then the next value, a distance h to the right, is created, f is evaluated there, and added to the previous value, and so forth. In this way a sum of functional values is created. Finally, the sum is multiplied by h, giving the sum of the areas of the rectangles, and we are done. X is purged to keep the menu clean.

Now we can create the Riemann sum rules very easily. The left endpoint rule starts with the number a, so the following program is all that is needed:

```
« A SUM »
```

We will store this program under the label LER, for Left Endpoint Rule.

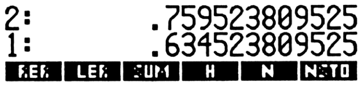
The right endpoint rule starts with the point a + h, so the following program suffices:

```
« A H + SUM »
```

We will store this program under the label RER.

All things should be ready for us to test these programs out now. Pressing

LER and RER



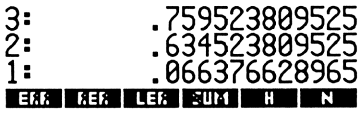
```
2:        .759523809525
1:        .634523809525
 RER | LER | SUM | H | N | NSTO
```

will give us two estimates for $\int_1^2 \frac{1}{x} dx$, using n = 4 subintervals.

Error Analysis

We know the actual value of this integral, namely ln 2. Let's do a little error analysis. To make it easy, we create a little program that immediately shows us how much error there is in our estimates. Store the program « 2 LN - » under the label ERR. Then pressing

LER ERR

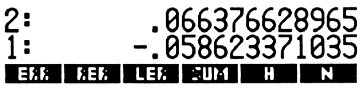


```
3:        .759523809525
2:        .634523809525
1:        .066376628965
 ERR | RER | LER | SUM | H | N
```

gives us the error in the left endpoint rule using the current value of n.

First of all, let's see what effect changing n has on the error. With n = 4, press

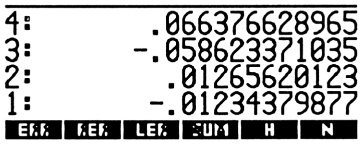
LER ERR and then RER ERR.



```
2:        .066376628965
1:       -.058623371035
 ERR | RER | LER | SUM | H | N
```

Then we will multiply n by some factor, say 5. That makes n = 20, so type

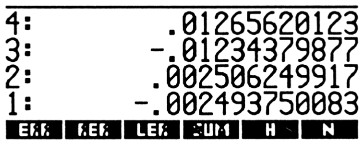
20 NSTO LER ERR RER ERR.



```
4:        .066376628965
3:       -.058623371035
2:          .01265620123
1:         -.01234379877
 ERR | RER | LER | SUM | H | N
```

Compare these errors with the previous errors. How are they related? Now let's multiply n by the factor 5 again, making n = 100. Type

100 NSTO LER ERR RER ERR.



```
4:          .01265620123
3:         -.01234379877
2:        .002506249917
1:       -.002493750083
 ERR | RER | LER | SUM | H | N
```

Compare again. What do you think?

You should have concluded that multiplying n by 5 divides the error by 5, approximately. In terms of accuracy, if a given n produces one decimal place of accuracy, it will take an n 10 times as large to produce two decimal places of accuracy.

This brings up the question of whether we can get whatever accuracy we like by increasing n. Let's investigate. With n = 4, LER and RER both gave us one decimal place of accuracy, if we rounded off. Try these again, and estimate the time it takes for the calculator to run the programs. I get about a second. So let's suppose that we can get one decimal place of accuracy in one second. That means it will take 10 seconds to get two decimal places of accuracy, 100 seconds to get three, and so forth. What if we want 12 decimal places of accuracy? That requires 10^{12} seconds. Use your calculator to turn that into years. Discouraging, isn't it?

What this little exercise demonstrates is that something so simple as the Riemann sum rules will never give us very much accuracy. We have to be smarter than that.

The integrand $f(x) = \frac{1}{x}$ has a fairly flat graph, meaning that rectangles do a fair job of approximating the function. If the integrand had a steeper graph, it is clear that rectangles would do a poorer job yet. Thus the error in the Riemann sum rules also depends on the steepness of the graph, which is given by the derivative of the function. Textbooks give the following error estimate: If $|f'(x)| \le B$ on [a, b], then the error in approximating $\int_a^b f(x)\,dx$ by a Riemann sum rule is no greater than $\frac{B(b-a)^2}{2n}$.

Trapezoidal Rule

$\boxed{7.8}$

You probably noticed when computing errors above that the LER and RER programs had about the same amount of error, but in opposite directions. That is, one was an overestimate, and the other, an underestimate. This makes us wonder what would happen if we averaged the two estimates; will the error cancel out?

Let's try it. Store the program

```
« LER RER + 2 / »
```

under the label TRAP. This is the trapezoidal rule. Geometrically, the rectangles on the subintervals have been replaced by trapezoids. Try TRAP a few times, and try to determine how multiplying n by a factor affects the error.

You should discover that multiplying n by a factor of k divides the error by a factor of about k^2. Thus, if a given value of n gives one decimal place of accuracy, then about 3 times n will give two decimal places of accuracy. (Here, $3 \approx \sqrt{10}$.)

It is also evident that the concavity of the integrand affects the accuracy of the trapezoidal rule, and that is why the second derivative figures into the error formula:

If $|f''(x)| \leq B$ on $[a, b]$, then the error in approximating $\int_a^b f(x)\,dx$ by the trapezoidal rule is no greater than $\frac{B(b-a)^3}{12n^2}$.

Midpoint Rule

Another obvious way to improve on the left and right endpoint rules is to use the midpoint of each interval rather than an endpoint. This gives the midpoint rule, whose program is

```
    « H 2 / A + SUM »
```

Store this under the label MPR. Experiment with MPR a few times, to see how multiplying n by a factor affects the error. You should discover the same "quadratic" law that was the case with the trapezoidal rule.

Simpson's Rule

Since TRAP and MRR have similar error behavior, we wonder if combining them could reduce the error still further. When we use both MPR and TRAP, we notice that the error of TRAP is about twice that of MPR, and in the opposite direction. This suggests a weighted average, as given by the program

```
« MPR 2 * TRAP + 3 / »
```

Store this under the label SIMP. That's right, it is Simpson's rule. Use SIMP a few times, to see how the error behaves when n is multiplied by a factor.

You should discover that multiplying n by k divides the error by about k^4. One reason for this is that, by using the midpoints of the subintervals also, we have essentially doubled the number of subintervals. The textbook error formula for the above program is as follows.

If $|f^{(4)}(x)| \leq B$ on $[a, b]$, then the error in approximating $\int_a^b f(x)\,dx$ by a Riemann sum rule is no greater than $\frac{B(b-a)^5}{2880n^4}$. The influence of the fourth derivative instead of the third is a surprise.

The complete menu under N.INT is now

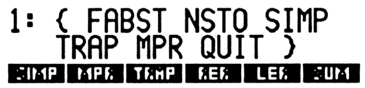
```
SIMP     MPR      TRAP     ERR      RER      LER
H        N        NSTO     F        A        B
FABST    QUIT
```

It might be more convenient to reorder this menu, so that FABST and NSTO are nearer to the front. Purge the program ERR and order the menu as follows:

```
FABST    NSTO     SIMP     TRAP     MPR      QUIT
RER      LER      H        N        F        A
B
```

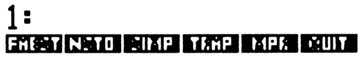
This is done by creating the list

{ FABST NSTO SIMP TRAP MPR QUIT }



```
1: { FABST NSTO SIMP
   TRAP MPR QUIT }
SIMP MPR TRAP REC LEC SUM
```

and then typing O R D E R $\boxed{\text{ENTER}}$.



```
1:
FABST NSTO SIMP TRAP MPR QUIT
```

Chapter 8. Sequences and Series

Some aspects of the subjects of sequences and series lend themselves to computation. A few ways in which the HP symbol-manipulating calculators are useful are presented here.

Terms of a Sequence

$\boxed{9.1}$

If the terms of a sequence are defined by a formula, then the formula can be evaluated just as a function can be, using the Solver, as discussed in Chapter 2. Thus, as many specified terms of the sequence can be found as desired, just by evaluating.

Sequences by Formula

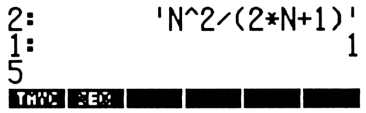
A simple program can be written to create specified terms of a sequence, as well. If the formula, the number of the first term desired, and the number of the last term are supplied, the following program, called SEQ, will create a list of the terms specified.

```
« → A N1 N2 « N1 N2 FOR K K 'N' STO A EVAL NEXT N2 N1
- 1 + →LIST 'N' PURGE » »
```

28S checksum: "E3DC"
48SX checksum: # 30925d

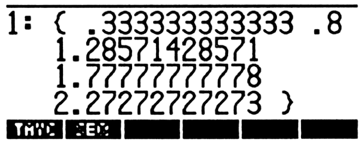
For example, to create the first five terms of the sequence $\left\{\frac{n^2}{2n + 1}\right\}$, type

'N ^ 2 / (2 * N + 1 $\boxed{\text{ENTER}}$ 1 $\boxed{\text{ENTER}}$ 5



```
2:        'N^2/(2*N+1)'
1:                    1
5
```

$\boxed{\text{SEQ}}$.



```
1: { .333333333333 .8
   1.28571428571
   1.77777777778
   2.27272727273 }
```

If you want the terms of the sequence in fraction form, insert the command →Q into the program SEQ just before the command NEXT.

Recursive Sequences

If a sequence is defined recursively, then a program can be written to produce the next term, given the preceding terms on which it depends.

For example, the Fibonacci sequence is defined by the recursion $a_1 = 0$, $a_2 = 1$, and $a_{n+2} = a_n + a_{n+1}$ for any n. The following program, called FIB, will produce the next term, given any two consecutive terms of the Fibonacci sequence.

```
« DUP2 + »
```

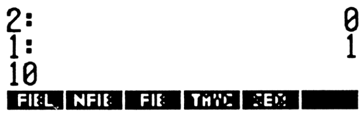
For example, with 0 in level 2 and 1 in level 1, pressing $\boxed{\text{FIB}}$ will produce the next term, 1, in level 1, moving the others up. Repeated applications of FIB produce successive terms of the sequence.

To automate the application of FIB, the following program, called NFIB, will take two terms and the number n of new terms desired from the stack and return the next n terms to the stack.

```
« 1 SWAP START FIB NEXT »
```

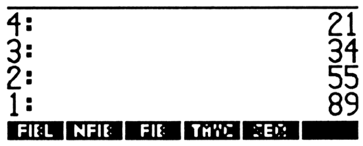
For example, to get the first twelve terms of the Fibonacci sequence, type

0 $\boxed{\text{ENTER}}$ 1 $\boxed{\text{ENTER}}$ 10



```
2:                  0
1:                  1
10
```

$\boxed{\text{NFIB}}$.



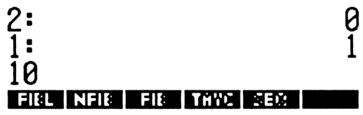
```
4:                 21
3:                 34
2:                 55
1:                 89
```

If you want a list of the next n terms of the sequence, the following program, FIBL, will give it to you.

```
« → N « N NFIB N 2 + →LIST » »
```

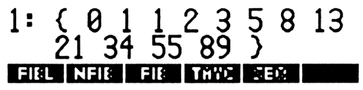
Type

0 $\boxed{\text{ENTER}}$ 1 $\boxed{\text{ENTER}}$ 10



```
2:                  0
1:                  1
10
```

$\boxed{\text{FIBL}}$.



```
1: { 0 1 1 2 3 5 8 13
   21 34 55 89 }
```

The above techniques used to produce the Fibonacci sequence will work with any recursively-defined sequence. In fact, the programs NBISC and NNWT, for approximating zeros of a function by the bisection method or by Newton's method, respectively, are just programs for producing some more terms of a sequence. You must determine the exact nature of the programs to use each time, but these examples make the process evident.

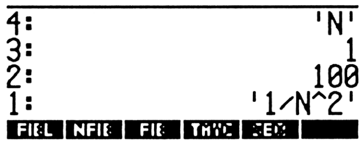
Partial Sums

$\boxed{9.2}$

The nth partial sum of a series is just the sum of the first n terms of the series. The HP48SX has a built-in command, Σ, for computing a partial sum. This is described and illustrated on pp. 423-426 of the *HP-48SX Owner's Manual.*

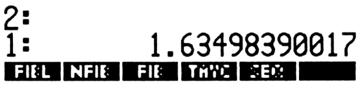
For example, to compute the sum $\sum_{n=1}^{100} \frac{1}{n^2}$, we just enter the sum and evaluate it. The *Owner's Manual* illustrates using the EquationWriter environment, so we will illustrate using the stack. Type

' N $\boxed{\text{ENTER}}$ 1 $\boxed{\text{ENTER}}$ 100 $\boxed{\text{ENTER}}$
' 1 / N ^ 2 $\boxed{\text{ENTER}}$



```
4:                'N'
3:                  1
2:                100
1:           '1/N^2'
```

$\boxed{\Sigma}$.



```
2:
1:    1.63498390017
```

For the HP-28S, we can use the idea of the program SEQ above to compute the partial sum; instead of making a list of the terms we compute, we add them up. The following program, PSUM, does the trick.

```
« → A M « 0 1 M FOR K K 'N' STO A EVAL + NEXT » 'N'
PURGE »
```
28S checksum: "82E6"

This program takes the formula for the nth term from level 2 and the number m of terms to use from level 1, and returns the mth partial sum. It assumes that N is the index of summation.

For example, to find the partial sum $\sum_{n=1}^{100} \frac{1}{n^2}$, type ' 1 / N ^ 2 $\boxed{\text{ENTER}}$ 100 $\boxed{\text{PSUM}}$.

Geometric Series

A geometric series is a series of the form $\sum_{n=0}^{\infty} ar^n$. If $|r| < 1$, then the series converges, and has sum $\frac{a}{1 - r}$. Thus, for geometric series, we can find not only terms and partial sums, as above, but also the sum. The environment of the Solver, with the formula $\frac{a}{1 - r}$, provides an excellent setting for finding the sums of various geometric series.

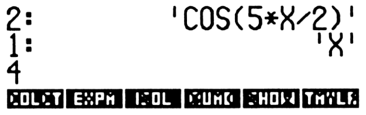
Taylor Polynomials

$\boxed{9.6}$

The HP-28S and HP-48SX have built into them the means for computing Taylor polynomials of functions. The command TAYLR on the ALGEBRA menu takes a function, the independent variable, and the degree from the stack, and returns the Taylor polynomial of the function of the specified degree in the independent variable, centered at zero.

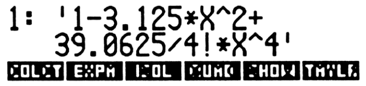
For example, to compute the Taylor polynomial of f(x) = cos $\frac{5x}{2}$ of degree 4 about 0, type

' C O S (5 * X / 2 $\boxed{\text{ENTER}}$ ' X $\boxed{\text{ENTER}}$ 4



```
2:           'COS(5*X/2)'
1:                   'X'
4
```

$\boxed{\text{TAYLR}}$.



```
1: '1-3.125*X^2+
   39.0625/4!*X^4'
```

You may find it interesting to apply the command $\boxed{\to\text{Q}}$ or $\boxed{\text{COLCT}}$ $\boxed{\to\text{Q}}$ afterward.

Since the Taylor polynomials computed by the calculator are centered at zero, they are partial sums of Maclaurin series. Thus the TAYLR command can be interpreted as creating the specified number of terms of the Maclaurin series of the function given.

If you want to compute a Taylor polynomial about some other point c, the substitution of y + c for x before the computation and x - c for y afterward will do it. For example, to find the Taylor polynomial of f(x) = cos $\frac{5x}{2}$ of degree 4 about 1, type ' C O S (5 * X / 2 $\boxed{\text{ENTER}}$ ' Y + 1 $\boxed{\text{ENTER}}$ ' X $\boxed{\text{STO}}$ $\boxed{\text{EVAL}}$ ' Y $\boxed{\text{ENTER}}$ 4 $\boxed{\text{TAYLR}}$ ' X $\boxed{\text{PURGE}}$ ' X - 1 $\boxed{\text{ENTER}}$ ' Y $\boxed{\text{STO}}$ $\boxed{\text{EVAL}}$ ' Y $\boxed{\text{PURGE}}$.

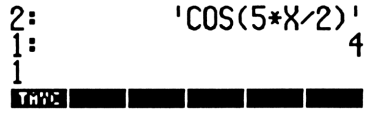
This can be automated, of course; consider the following program, called TAYC.

```
« → N C « 'Y' C + 'X' STO EVAL 'Y' N TAYLR 'X' PURGE 'X'
C - 'Y' STO EVAL 'Y' PURGE » »
                                        28S checksum: "68CB"
                                        48SX checksum: # 35391d
```

This program takes the function f, written in terms of the variable X, from level 3, the degree n from level 2, and the center c from level 1, and returns the Taylor polynomial of degree n for f centered at c.

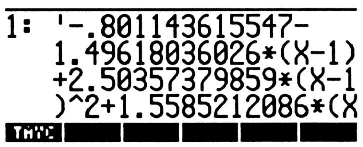
For example, to redo the above example, and get the Taylor polynomial of f(x) = cos $\frac{5x}{2}$ of degree 4 about c = 1, type

' C O S (5 * X / 2 [ENTER] 4 [ENTER] 1



```
2:        'COS(5*X/2)'
1:                   4
1
```

[TAYC].



```
1: '-.801143615547-
   1.49618036026*(X-1)
   +2.50357379859*(X-1
   )^2+1.5585212086*(X
```

If the function given involves other variables than the one specified for use with TAYLR, the polynomial's coefficients are given in terms of those variables. For use with TAYC, the function given can involve other variables except for Y, and must use X as the variable supplied to TAYLR.

Approximate Symbolic Integration

$\boxed{\boxed{7.1\text{-}7.7}}$

It was mentioned at the end of Chapter 4 that the HP-28S would find the antiderivative of a polynomial, but not of other functions. When you ask the HP-28S to find the antiderivative of any function, what it actually does is to compute the Taylor polynomial of the function of the degree specified and then return the antiderivative of the Taylor polynomial. Since the Taylor polynomial of a polynomial function is itself, this works quite well for finding the antiderivative of a polynomial, but only "approximately" for non-polynomial functions.

Therefore, if you wish to find the antiderivative of a function f(x), you must not only give the function and name the independent variable, but also indicate the degree of the Taylor polynomial to be used. For a polynomial function, it is sufficient to give the degree of the polynomial. In every case, the calculator leaves to the user the supplying of the constant of integration.

For example, if you want to find the antiderivative of the function f(x) = cos x, you would type something like ' C O S (X [ENTER] ' X [ENTER] 4 [ENTER] ∫ . The calculator will return the Taylor polynomial of degree 5 for the sine function, rather than the sine function itself. It is easy to see how the symbolic integration can be very misleading if the user is unaware of what is actually taking place.

Chapter 9. Conic Sections and Polar Coordinates

The HP symbol-manipulating calculators can be programmed to produce graphs of conic sections and other implicitly-defined functions, and of parametric equations. They also have the means built in for handling polar coordinates in an efficient fashion.

Graphs of Conics

$\boxed{10.2\text{-}10.6}$

The conic sections, with equations of the form

$$ax^2 + bxy + cy^2 + dx + ey + f = 0,$$

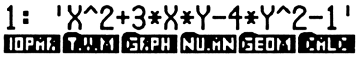
are special cases of implicitly-defined functions, with equations of the form

$$f(x, y) = 0.$$

The HP-48SX has built-in an environment for plotting the graphs of implicit functions, which it interestingly enough calls the CONIC type. Type in the function f(x, y) and store it as the equation by pressing STEQ on the PLOT menu. Then specify CONIC type by pressing CONIC on the PLOT PTYPE menu. Then draw the graph by pressing DRAW on the PLOT PLOTR menu. (ERASE the previous graph first if you wish.)

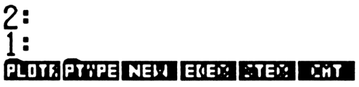
For example, to get the graph of $x^2 + 3xy - 4y^2 = 1$, type

'X ^ 2 + 3 * X * Y - 4 * Y ^ 2 - 1 ENTER



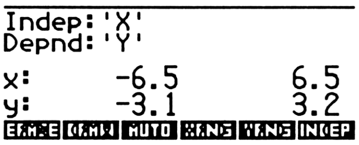
```
1:  'X^2+3*X*Y-4*Y^2-1'
```

PLOT PTYPE CONIC STEQ



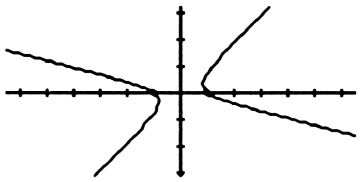
```
2:
1:
```

PLOTR ERASE



```
Indep: 'X'
Depnd: 'Y'
x:      -6.5        6.5
y:      -3.1        3.2
```

DRAW.



On the HP-28S, the DRAW command can use a function with one variable only, so we need a substitute for DRAW for implicit functions. Here is one, called IMPG, that can be substituted into the programs of Chapter 3.

```
« DRAX PPAR LIST→ 4 DROPN C→R 'Y2' STO 'X2' STO C→R 'Y1'
STO 'X1' STO X2 X1 - 5 * 136 / 'DX' STO Y2 Y1 - 5 * 31 /
'DY' STO Y1 'Y' STO 1 6 START X1 'X' STO EQ →NUM 'Z1' STO
DY 'Y' STO+ EQ →NUM 'Z2' STO 1 27 START DX 'X' STO+ EQ
→NUM 'Z4' STO Y DY - 'Y' STO EQ →NUM 'Z3' STO IF Z1 Z2 *
0 < Z1 Z3 * 0 < OR Z1 Z4 * 0 < OR THEN 1 5 FOR K X DX -
DX 5 / K * + 'T' STO Z3 Z1 - K * 5 / Z1 + 'T1' STO Z4 Z2 -
K * 5 / Z2 + 'T2' STO IF T1 T2 * 0 < THEN T DY T1 * T1 T2
- / Y + R→C PIXEL END NEXT 1 5 FOR J Y DY 5 / J * + 'T'
STO Z2 Z1 - J * 5 / Z1 + 'T1' STO Z4 Z3 - J * 5 / Z3 + 'T2'
STO IF T1 T2 * 0 < THEN X DX - DX T1 * T1 T2 - / + T R→C
PIXEL END NEXT END Z4 'Z2' STO Z3 'Z1' STO DY 'Y' STO+
NEXT NEXT { X1 X2 Y1 Y2 DX DY Z1 Z2 Z3 Z4 T T1 T2 X Y }
PURGE »
                                        28S checksum: "B81D"
```

The above program is based on an interpolation routine that tests for sign changes in small boxes, skipping the interiors of the boxes that have no sign changes in order to save time. Where sign changes occur, every pixel is tested for the sign change, and an unbroken graph is created.

For the HP-28S, it is recommended that a subdirectory IMP.G be created, containing the following programs. They are all described in Chapter 3, except for IMPG itself.

CLDRA	OVDRA	RCLGR	NWIN	REDRA	QUIT
EQ	ZIN	ZOUT	EQS	SCR	IMPG
PPAR	EQ				

In each of the programs CLDRA, OVDRA, and REDRA, IMPG would replace DRAW wherever it is mentioned. This environment will handle the graphing of implicit functions and also contour maps, as we will see later.

As an example, if we wish to plot the graph of $x^2 + 3xy - 4y^2 = 1$, we would type in ' X ^ 2 + 3 * X * Y - 4 * Y ^ 2 - 1 $\boxed{\text{ENTER}}$ and then press $\boxed{\text{CLDRA}}$. We would have to adjust the plot parameters, perhaps, in order to see the portion of the curve we wish; the programs BOX, ZIN, and ZOUT are useful in that regard. In this particular case, the default parameters give a very satisfactory picture of the hyperbola. After pressing $\boxed{\text{ON}}$ to return to the stack display, pressing $\boxed{\text{RCLGR}}$ will instantly return the graph to the display.

Rotation of the Coordinate System

$\boxed{10.5}$

In the equation

$$ax^2 + bxy + cy^2 + dx + ey + f = 0$$

of a conic section, the xy-term can be eliminated by a rotation of the coordinate system. The angle through which the coordinate system should be rotated is given by

$$\cot 2\alpha = \frac{a - c}{b},$$

and the rotation equations

$$x = u \cos \alpha - v \sin \alpha$$
$$y = u \sin \alpha + v \cos \alpha,$$

when substituted for x and y in the original equation, give the new equation of the conic section in the uv-coordinate system. The process of carrying out these operations can be very tedious. Using the symbol-manipulation power of the HP calculators, this process can be reduced to a program.

The following program, called ROTCO, for "ROTate COnic", takes the equation of a conic from level 4 and the coefficients a, b, and c from levels 3, 2, and 1, and returns the "simplified" equation in the uv-coordinate system. This program also uses the program EXCO, given in the *HP-28S Owner's Manual*, starting on p. 253, and in the *HP-48SX Owner's Manual*, starting on p. 569.

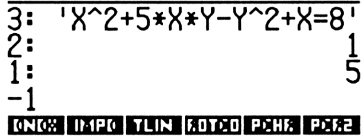
```
« ROT SWAP - SWAP / DUP 0 IF == THEN DROP π 4 / →NUM
ELSE INV ATAN 2 / END → T « 'U' T COS * 'V' T SIN * - 'X'
STO 'U' T SIN * 'V' T COS * + 'Y' STO EVAL EXCO » { X Y }
PURGE »
```

28S checksum: "3DEB"
48SX checksum: # 43082d

The equation of the conic should be entered in terms of X and Y, and the coefficients a, b, and c must be entered in order. If b = 0, the rotation is unnecessary, and the attempt to use the program will result in a "division by zero" error. If a uv-term appears in the result, it is because of round-off error, and it can be eliminated.

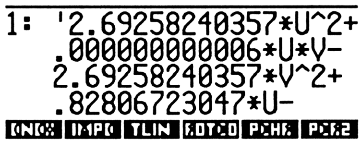
For example, to eliminate the xy-term in the equation $x^2 + 5xy - y^2 + x = 8$, we type

'X ^ 2 + 5 * X * Y - Y ^ 2 + X = 8 $\boxed{\text{ENTER}}$
1 $\boxed{\text{ENTER}}$ 5 $\boxed{\text{ENTER}}$ 1 $\boxed{\text{CHS}}$



```
3:  'X^2+5*X*Y-Y^2+X=8'
2:                    1
1:                    5
-1
```

$\boxed{\text{ROTCO}}$.



```
1:  '2.69258240357*U^2+
    .000000000006*U*V-
    2.69258240357*V^2+
    .82806723047*U-
```

The program takes a few moments (it's the EXCO part), so be patient.

Parametric Equations

$\boxed{11.1}$

A pair of parametric equations

$$\begin{cases} x = f(t) \\ y = g(t) \end{cases}, t \in [a, b],$$

define a curve in the plane from (f(a), g(a)) to (f(b), g(b)), provided that f and g are continuous functions. The HP calculators can help us find the slope of such a curve, and sketch its graph.

Chain Rule

The parametric form of the chain rule is

$$\frac{dy}{dx} = \frac{dy/dt}{dx/dt} = \frac{g'(t)}{f'(t)}$$

and gives the slope of the parametric curve. The following little program for the HP takes the functions f and g from the stack and returns the derivative dy/dx. The name of this program is PCHR, for "Parametric CHain Rule."

```
« 'T' ∂ SWAP 'T' ∂ / »
```

28S checksum: "2FA"
48SX checksum: # 44999d

The functions for x and y should be entered in terms of T, and in the order x, then y.

For example, if the parametric equations are

$$\begin{cases} x = \sin t \\ y = t^3 \end{cases},$$

then to find dy/dx we type

' S I N (T [ENTER] ' T ^ 3 [ENTER]

```
2:                'SIN(T)'
1:                  'T^3'
CNCS IMPC TLIN ROTCO PCHR PCR2
```

[PCHR].

```
1:         '3*T^2/COS(T)'
CNCS IMPC TLIN ROTCO PCHR PCR2
```

We can even write a program to find the second derivative in parametric equations. If we let $y' = \frac{dy}{dx}$, then $\frac{d^2y}{dx^2} = \frac{dy'/dt}{dx/dt}$. The following program, called PCR2 (for "Parametric Chair Rule, 2nd derivative"), will then do the trick. Just like PCHR, it starts with the parametric functions f and g on the stack.

```
« SWAP DUP → F « SWAP PCHR F SWAP PCHR » »
                                    28S checksum: "7D79"
                                    48SX checksum: # 11824d
```

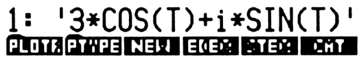
You may want to use EXCO to simplify the result.

Graphs

The HP-48SX has built-in a procedure for drawing the graph of a pair of parametric equations, as described on p. 332 of the HP-48SX Owner's Manual. You must specify PARAMETRIC plot type by pressing $\boxed{\text{PARA}}$ on the PLOT PTYPE menu. Then write the functions x = f(t) and y = g(t) in the form f(t) + ig(t), a complex number with real part f(t) and imaginary part g(t). Store f(t) + ig(t) as the equation by pressing $\boxed{\text{STEQ}}$ on the PLOT menu. Finally, specify the interval over which t is to vary; if t runs from a to b, type the list { T a b } and press $\boxed{\text{INDEP}}$ on the PLOT PLOTR menu. Then press $\boxed{\text{DRAW}}$.

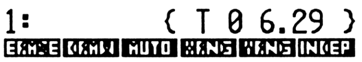
For example, to get the graph of the pair of equations x = 3 cos t, y = sin t, with t running from 0 to 2π, type

$\boxed{\text{PLOT}}\boxed{\text{PTYPE}}$ $\boxed{\text{PARA}}$ ' 3 * $\boxed{\text{COS}}$ T
$\boxed{\blacktriangleright}$ + i * $\boxed{\text{SIN}}$ T $\boxed{\text{ENTER}}$



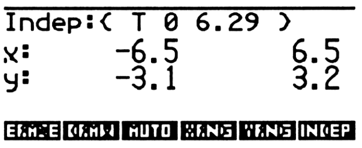
```
1: '3*COS(T)+i*SIN(T)'
```

$\boxed{\text{STEQ}}$ $\boxed{\text{PLOTR}}$ { T 0 6.29 $\boxed{\text{ENTER}}$



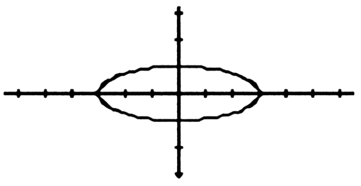
```
1:        { T 0 6.29 }
```

$\boxed{\text{INDEP}}$



```
Indep:{ T 0 6.29 }
x:     -6.5        6.5
y:     -3.1        3.2
```

$\boxed{\text{ERASE}}$ $\boxed{\text{DRAW}}$.



For the HP-28S, it is probably best to create an environment for parametric equations in a separate subdirectory PAR.G. The complete menu under PAR.G is

XSTO	YSTO	ABSTO	CLDRA	RCLGR	QUIT
X	Y	A	B	N	SCR
PPAR					

This environment can be embellished by adding programs for overdrawing and so forth, as in Chapter 3. Here is a short description of these menu items, mentioned in reverse order for convenience in entering.

• PPAR contains the current plot parameters.

• SCR contains the most recently created screen.

• N is the number of pixels to be plotted; 50 or 100 is usually a good number.

• B is the upper limit of T, and is created by the program ABSTO below. Store a zero here now to get B on the menu in the right place.

• A is the lower limit of T, and is also created by ABSTO.

• Y contains the function y = g(t), and is created by the program YSTO below.

• X contains the function x = f(t), and is created by XSTO.

• QUIT is just the program « UP », assuming that UP is on the top level of the USER menu.

• RCLGR is the program « SCR →LCD DGTIZ ».

• CLDRA is the main program, and actually does the plotting. It is

```
« B A - N / →NUM → H « CLLCD DRAX A 'T' STO 1 N START
X →NUM Y →NUM R→C PIXEL H 'T' STO+ NEXT 'T' PURGE
LCD→ 'SCR' STO » DGTIZ »
                                        28S checksum: "529B"
```

• ABSTO is the program « 'B' STO 'A' STO », and is used to store the limits A and B of the parameter T.

• YSTO is the program « 'Y' STO », and is used to store the function y = g(t). The function should be entered in algebraic form, using T as the variable.

• XSTO is the program « 'X' STO », and stores the function x = f(t).

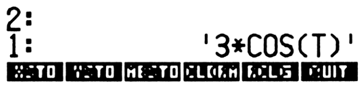
It is well to remember that the graph produced here depends as much on the range of T as on the plot parameters. Plot parameters determine the window, but the limits A and B determine how much of the graph is drawn, and N determines how many pixels are set in the graph. Only window information can be obtained from digitized points; you must adjust A, B, and N "by hand."

To illustrate the use of this environment, suppose we wish to plot the graph of the parametric equations

$$\begin{cases} x = 3 \cos t \\ y = 2 \sin t \end{cases}, t \in [0, 2\pi],$$

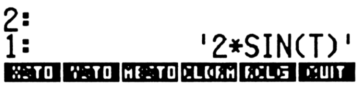
plotting 75 pixels. We type

' 3 * C O S (T $\boxed{\text{ENTER}}$



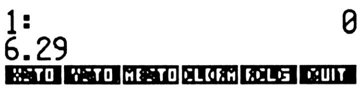
```
2:
1:              '3*COS(T)'
```

$\boxed{\text{XSTO}}$ ' 2 * S I N (T $\boxed{\text{ENTER}}$

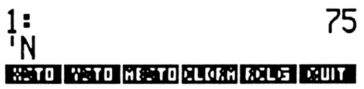


```
2:
1:              '2*SIN(T)'
```

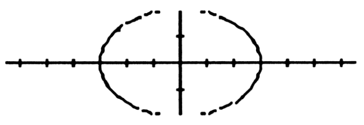
$\boxed{\text{YSTO}}$ 0 $\boxed{\text{ENTER}}$ 6.29


```
1:                 0
6.29
```

$\boxed{\text{ABSTO}}$ 75 $\boxed{\text{ENTER}}$ ' N


```
1:                75
'N
```

$\boxed{\text{STO}}$ $\boxed{\text{CLDRA}}$.



We may wish to adjust plot parameters to see all of the picture. The values and functions in X, Y, A, B, and N can be adjusted independently for subsequent graphs. Their values will not change until you change them.

Polar Coordinates

$\boxed{\text{11.2,11.3}}$

Routines for handling polar coordinates and transforming to rectangular coordinates and back again are built into the HP calculators. This makes both computation and graphing easy to do.

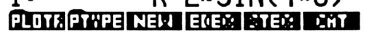
The transformation from rectangular to polar coordinates and back again was discussed in Chapter 2 when we dealt with vectors. Further discussion is found on p. 277 of the *HP-28S Reference Manual* and on pp. 169ff of the *HP-48SX Owner's Manual.*

Graphs

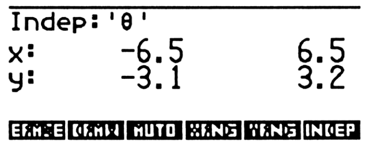
The HP-48SX has a built-in procedure for plotting the graphs of functions in polar coordinates, as described beginning on p. 330 of the HP-48SX Owner's Manual. First, make sure that the calculator is in radian mode. Then specify POLAR type by pressing $\boxed{\text{POLAR}}$ on the PLOT PTYPE menu. Then type in the equation r = f(θ) and store it as the equation. Then set θ to be the independent variable, and DRAW it. (The symbol θ is a right-shifted F in α mode.) If you do not specify a range for the independent variable, the interval [0, 2π] is chosen automatically. To specify a range [c, d], designate the list { θ c d } as the independent variable.

For example, to plot the graph of r = 2 sin 4θ, type

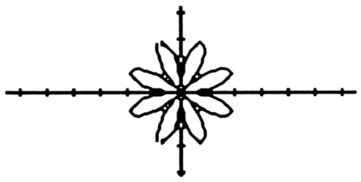
$\boxed{\text{PLOT}}$ $\boxed{\text{PTYPE}}$ $\boxed{\text{POLAR}}$ ' R = 2 * $\boxed{\text{SIN}}$ 4 * θ $\boxed{\text{ENTER}}$ (θ is α right-shift F)


```
1:      'R=2*SIN(4*θ)'
```

$\boxed{\text{STEQ}}$ $\boxed{\text{PLOTR}}$ θ $\boxed{\text{INDEP}}$


```
Indep: 'θ'
x:      -6.5        6.5
y:      -3.1        3.2
```

$\boxed{\text{ERASE}}$ $\boxed{\text{DRAW}}$.



For the HP-28S, it is fairly easy to get graphs in polar coordinates. Here is a subdirectory POL.G which is an environment for plotting the graphs of polar functions of the form $r = f(\theta)$. The complete menu under POL.G is

RSTO	ABSTO	CLDRA	RCLGR	R	QUIT
A	B	N	SCR	PPAR	

This environment can be enhanced by adding programs for overdrawing and so forth, as in Chapter 3. Here is a short description of these menu items:

• PPAR contains the current plot parameters.

• SCR contains the current screen.

• N is the number of pixels in the plot.

• B is the upper limit of θ.

• A is the lower limit of θ.

• R is the function $r = f(\theta)$.

• RCLGR is the program « SCR →LCD DGTIZ »

CLDRA is the program

```
        « B →NUM A →NUM - N / 'DT' STO A 'T' STO CLLCD DRAX 1 N
START R EVAL T R→C P→R PIXEL DT 'T' STO+ NEXT { T DT } PURGE
LCD→ 'SCR' STO DGTIZ »
                                               28S checksum: "8700"
```

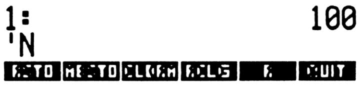
• ABSTO is the program « 'B' STO 'A' STO ».

• RSTO is the program « 'R' STO ».

The function $r = f(\theta)$ is entered in algebraic form, using T instead of θ. As for parametric equations, the plot parameters determine the window, but the values of A, B, and N determine how much of the graph is drawn.

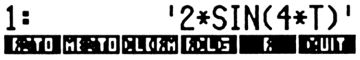
To illustrate, suppose we first want to plot the graph of $r = 2 \sin 4\theta$ from 0 to 2π, using 100 points on a graph. We type

100 $\boxed{\text{ENTER}}$ 'N



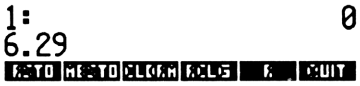
```
1:                 100
'N
```

$\boxed{\text{STO}}$ ' 2 * S I N (4 * T $\boxed{\text{ENTER}}$



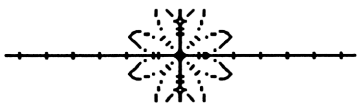
```
1:        '2*SIN(4*T)'
```

$\boxed{\text{RSTO}}$ 0 $\boxed{\text{ENTER}}$ 6.29



```
1:                   0
6.29
```

$\boxed{\text{ABSTO}}$ $\boxed{\text{CLDRA}}$.



We may want to adjust the plot parameters to see the graph better.

Chapter 10. Solid Analytic Geometry

Points and Planes; Vector Methods

12.1,12.4,12.5

In three dimensions, a point has three coordinates, and can be represented readily to the HP calculator as a vector. That is, the point (x, y, z) can be given as the array [x y z]. A plane in space has an equation of the form ax + by + cz + d = 0, and can be represented to the calculator as the vector [a b c d]. These ideas are used in the following little programs.

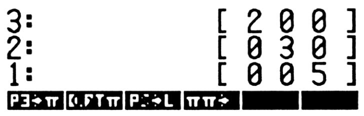
The Plane on Three Points

P3→π is a program that takes three points from the stack and returns the plane they determine.

```
« → U V W « U V - U W - CROSS → A « A 1 GET 'X' * A
2 GET 'Y' * + A 3 GET 'Z' * + A U DOT - 0 = » » »
                                        28S checksum: "52E1"
                                        48SX checksum: # 35634d
```

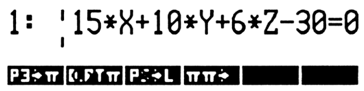
For example, to find the plane containing the three points (2, 0, 0), (0, 3, 0), and (0, 0, 5), type

[2 , 0 , 0 ENTER [0 , 3 , 0 ENTER
[0 , 0 , 5 ENTER



```
3:              [ 2 0 0 ]
2:              [ 0 3 0 ]
1:              [ 0 0 5 ]
```

P3→π .



```
1: '15*X+10*Y+6*Z-30=0
```

If the result is ever '0 = 0', it means that the three points given are collinear, and hence do not determine a plane.

Distance from a Point to a Plane

D.PTπ is a program that takes a point and a plane from the stack and returns the (shortest) distance between them. Both the point and plane are entered as vectors.

```
« → P PL « P ARRY→ DROP 1 4 →ARRY PL DOT ABS PL
ARRY→ DROP DROP 3 →ARRY ABS / » »
                                        28S checksum: "91A6"
```

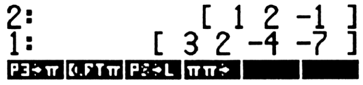
For the HP-48SX, the same program is

```
« → P PL « P OBJ→ DROP 1 4 →ARRY PL DOT ABS PL OBJ→
DROP DROP 3 →ARRY ABS / » »
```

48SX checksum: # 48188d

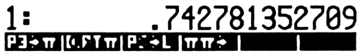
For example, to find the distance between the point (1, 2, -1) and the plane 3x + 2y - 4z - 7 = 0, type

[1 , 2 , 1 CHS ENTER [3 , 2 , 4 CHS , 7 CHS ENTER



```
2:          [ 1 2 -1 ]
1:       [ 3 2 -4 -7 ]
```

D.PTπ .



```
1:       .742781352709
```

Lines

A line in three dimensions can be represented most easily as a set of three linear parametric equations in the form

$$\begin{cases} x = x_0 + at \\ y = y_0 + bt \\ z = z_0 + ct \end{cases}$$

The following little programs create the parametric equations of lines.

Line on Two Points

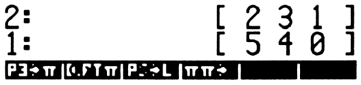
PS→L takes two points in space and gives the three parametric equations of the line they determine.

```
« → P1 P2 « P1 P2 - → D « 'X' P1 1 GET D 1 GET 'T' * + =
'Y' P1 2 GET D 2 GET 'T' * + = 'Z' P1 3 GET D 3 GET 'T' * +
= » » »
```

28S checksum: "4563"
48SX checksum: # 50865d

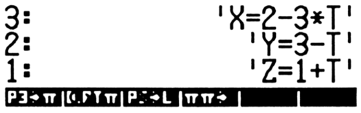
For example, to find the line on the points (2, 3, 1) and (5, 4, 0), type

[2 , 3 , 1 ENTER [5 , 4 , 0 ENTER



```
2:           [ 2 3 1 ]
1:           [ 5 4 0 ]
```

PS→L .



```
3:          'X=2-3*T'
2:            'Y=3-T'
1:            'Z=1+T'
```

Line of Intersection of Two Planes

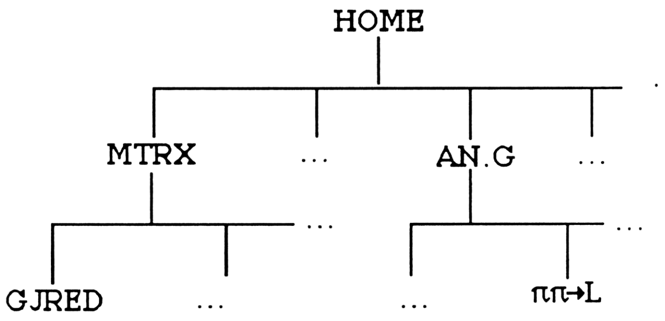
ππ→L takes two planes from the stack and returns the parametric equations of the line of intersection. The plane ax + by + cz + d = 0 is entered as the vector [a b c d].

```
« → P Q « P ARRY→ DROP DROP 3 →ARRY Q ARRY→ DROP
DROP 3 →ARRY CROSS → D « D ABS 0 IF == THEN "SAME OR
PARALLEL" ELSE P ARRY→ DROP NEG Q ARRY→ DROP NEG { 2
4 } →ARRY GJRED → A « 'A(1,1)' EVAL 0 IF == THEN 'X' D 1
GET 'T' * = 'Y' 'A(1,4)' EVAL D 2 GET 'T' * + = 'Z' 'A(2,4)'
EVAL D 3 GET 'T' * + = ELSE 'X' 'A(1,4)' EVAL D 1 GET 'T' *
+ = 'A(2,2)' EVAL 0 IF == THEN 'Y' D 2 GET 'T' * = 'Z' 'A(2,4)'
EVAL D 3 GET 'T' * + = ELSE 'Y' 'A(2,4)' EVAL D 2 GET 'T' *
+ = 'Z' D 3 GET 'T' * = END END » END » » »
                                        28S checksum: "37D5"
```

For the HP-48SX, the command ARRY→ in ππ→L is replaced by the command OBJ→ (in four occurrences). With that change, the 48SX checksum is # 33318d.

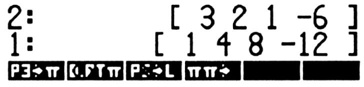
Notice that the above program uses the program GJRED, assuming it is in the same or a higher directory. If it is in a different subdirectory, then navigation directions to its location and back again must be given.

For example, if GJRED is in a subdirectory called MTRX and the program ππ→L is located in a subdirectory called AN.G, as in the diagram, then the command GJRED in the above program should be replaced with HOME MTRX GJRED HOME AN.G.



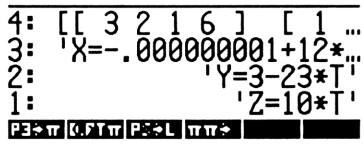
To use the program, for example, to find the line common to the planes 3x + 2y + z - 6 = 0 and x + 4y + 8z - 12 = 0, type

[3 , 2 , 1 , 6 $\boxed{\text{CHS}}$ $\boxed{\text{ENTER}}$ [1 , 4 , 8 , 12 $\boxed{\text{CHS}}$ $\boxed{\text{ENTER}}$



```
2:        [ 3 2 1 -6 ]
1:       [ 1 4 8 -12 ]
```

$\boxed{\pi\pi\to\text{L}}$.



```
4: [[ 3 2 1 6 ] [ 1 …
3: 'X=-.000000001+12*…
2:          'Y=3-23*T'
1:            'Z=10*T'
```

Space Curves and Graphs

$\boxed{13.1}$

A curve in space is most easily represented parametrically, as

$$\begin{cases} x = f(t) \\ y = g(t) \\ z = h(t) \end{cases}, \; t \in [a, b].$$

To sketch the graph of a curve in space requires what is called a *perspective transformation* to make the graph appear as though we were actually looking at it. This in turn requires that we specify the point from which we are looking, and then perform the perspective transformation on each point we plot.

Here is a subdirectory TH.D.G that is an environment for plotting space curves (and surfaces, which we will talk about later). It does so by performing the perspective transformation on the three spacial dimensions, so that the picture created on the screen represents the object as seen when looking toward the origin from a specified viewpoint (VX, VY, VZ) in space. Means are included to allow for magnification by a scale factor S.

The menu under TH.D.G is

```
        SPCRV     P.SRF     VSTO     SSTO     M        QUIT
        S         VX        VY       VZ       TRANS
```

• TRANS is a subroutine that performs the perspective transformation on a point of three-space. It uses the matrix M and the scale factor S that are supplied separately, as explained below. The input to TRANS is the set of coordinates x, y, z of a point in space.
The program is

```
« 1 4 →ARRY M SWAP * ARRY→ DROP DROP ROT ROT R→C S *
SWAP INV * »
                                        28S checksum: "C495"
```

On the HP-48SX, ARRY→ must be replaced by OBJ→. With this change, the 48SX checksum is # 46002d.

• VX, VY, and VZ are the coordinates of the viewpoint, and are created by VSTO.

• S is the scale factor, and is created by SSTO.

• QUIT is just « UP » (« UPDIR » on the HP-48SX), as you no doubt guessed.

• M is the matrix that performs most of the transformation, and is created by VSTO.

• SSTO is just the program « 'S' STO » for storing the scale factor S. I find that a value of about 20 for S gets most of the picture on the screen, depending on the function and on the viewing point. You will have to experiment a little.

• VSTO is a program for storing the coordinates of the viewpoint and creating the matrix M. It requires as input the coordinates VX, VY, and VZ (in that order). It is a good idea to select a viewpoint that will be outside the region of space in which the object to be viewed lies. The program is

```
« 'VZ' STO 'VY' STO 'VX' STO VX VY 2 →ARRY ABS 'D1' STO
D1 VZ 2 →ARRY ABS 'D2' STO CLΣ VY NEG D1 / VX D1 / 0 0 4
→ARRY Σ+ VX VZ * NEG D1 D2 * / VY VZ * NEG D1 D2 * / D1
D2 / 0 4 →ARRY Σ+ VX D2 / VY D2 / VZ D2 / D2 NEG 4
→ARRY NEG Σ+ 0 0 0 1 4 →ARRY Σ+ RCLΣ 'M' STO CLΣ { D1
D2 } PURGE »
                                        28S checksum: "71B8"
```

48SX checksum: # 26746d

The commands involving Σ in the above program are found on the STAT menu.

• P.SRF is a subdirectory for the plotting of surfaces, as will be explained later.

• SPCRV is a subdirectory for plotting the graph of the three parametric equations

$$\begin{cases} x = f(t) \\ y = g(t) \\ z = h(t) \end{cases}, t \in [a, b]$$

in space. The menu for SPCRV is

```
XYZST    ABSTO    DRAGR    RCLGR    E.OLD    QUIT
N        X        Y        Z        A        B
SCR      PPAR
```

• PPAR contains the plot parameters. Choice of plot parameters is much less important in this setting than the viewpoint and the scale factor S, so the default plot parameters should be fine.

• SCR is the container for storing the screen each time. This container is not needed on the HP-48SX.

• A and B are the limits on the parameter T, and are created by ABSTO.

• X, Y, and Z are the functions f(t), g(t), and h(t), and are stored by XYZST.

• N is the number of points that will be plotted, and must be stored separately.

• QUIT is the program « UP » or « UPDIR », as above.

• E.OLD is a program for erasing the old picture if you don't want the next picture drawn on top of it. For the HP-48SX, it is simply the program « ERASE ». For the HP-28S, it is the program « CLLCD LCD→ 'SCR' STO ». On the HP-28S, after you use E.OLD, you must press $\boxed{\text{ON}}$ to continue.

• RCLGR is a program for recalling the last picture drawn. On the HP-48SX, it is simply « GRAPH ». On the HP-28S, it is « SCR →LCD DGTIZ ».

• DRAGR is the program that does the plotting. For the HP-28S, it is

```
« B A - N / →NUM 'DT' STO A 'T' STO SCR →LCD 1 N START
X →NUM Y →NUM Z →NUM TRANS PIXEL T DT + 'T' STO NEXT
{ T DT } PURGE LCD→ 'SCR' STO »
```

28S checksum: "8A4"

For the HP-48SX, the program is

```
« B A - N / →NUM 'DT' STO A 'T' STO 1 N START X →NUM Y
→NUM Z →NUM TRANS PIXON T DT + 'T' STO NEXT { T DT }
PURGE GRAPH »
```

48SX checksum: # 22365d

• ABSTO is the program « 'B' STO 'A' STO ». It takes the limits A and B of T and stores them.

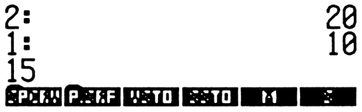
• XYZST takes the three functions f(t), g(t), and h(t) from the stack and stores them. The functions are entered in algebraic notation, using T as the variable. The program is just « 'Z' STO 'Y' STO 'X' STO ».

To use this environment, suppose we wish to plot the graph of

$$\begin{cases} x = \cos t \\ y = 3 \sin t \\ z = t/5 \end{cases}, t \in [0, 2\pi].$$

We must first choose a viewpoint, say (20, 10, 15). This we do by entering the subdirectory TH.D.G and typing

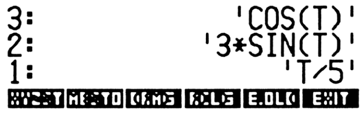
20 ENTER 10 ENTER 15



```
2:                20
1:                10
15
```

VSTO. We must also choose a scale constant S, and 40 is probably a good choice. Type 40 SSTO. Then enter the subdirectory SPCRV. If we want to plot 100 points, we type 100 ENTER 'N STO. To enter the parametric equations, we type

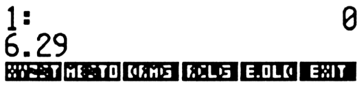
'COS(T ENTER '3*SIN(T ENTER 'T/5 ENTER



```
3:          'COS(T)'
2:        '3*SIN(T)'
1:             'T/5'
```

XYZST. The interval [0, 2π] is most easily entered by typing

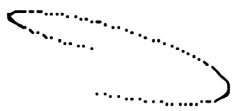
0 ENTER 6.29



```
1:                 0
6.29
```

ABSTO. Finally, plot the graph by pressing

DRAGR.



Parametric Surfaces

The various surfaces mentioned in calculus can all be graphed on the HP graphing calculators. This is often a slow and tedious job, even for the calculator, because so much computation is going on; some even suggest that it be left to larger computers entirely. However, it is fun to see what the calculator can do, so we present the material here for you to use if you want to.

We will describe all surfaces in terms of parametric surfaces. By way of definition, a parametric surface is the graph of the parametric equations

$$\begin{cases} x = f(u,v) \\ y = g(u,v) \\ z = h(u,v) \end{cases}, \quad u \in [a, b],\ v \in [c, d].$$

If a value of v is chosen and v is fixed at that value, then the equations become parametric equations in terms of the single parameter u, whose graph is a space curve; this curve is called a *u-curve* of the surface, and the set of all u-curves is called the *u-net* of the surface. Similarly, if the value of u is fixed, we get a *v-curve*, and the set of all v-curves is called the *v-net* of the surface.

By plotting a few curves of each net, a "wire-mesh" model of the surface is obtained; that is what we will mean by the graph of the surface. Before discussing the environment for plotting parametric surfaces on the HP calculators, we will describe some common surfaces in terms of parametric equations, and give a few examples.

Planes

$\boxed{12.5}$

A plane in space has an equation of the form $ax + by + cz + d = 0$. If the z-term is present ($c \neq 0$), then we may solve for z and get $z = -\frac{ax + by + d}{c}$. Therefore the parametric equations

$$\begin{cases} x = u \\ y = v \\ z = -\dfrac{au + bv + d}{c} \end{cases}$$

give us the plane. The u- and v-nets are lines in that plane parallel to the xz- and yz-planes.

If the z-term is absent, so that the equation of the plane is $ax + by + d = 0$, then the plane is vertical (parallel to the z-axis). If the y-term is present ($b \neq 0$), we may solve for y and get $y = -\frac{ax + d}{b}$, and the parametric equations

$$\begin{cases} x = u \\ y = -\dfrac{au + d}{b} \\ z = v \end{cases}$$

give us the plane. The u-net consists of horizontal lines, and the v-net consists of vertical lines.

If both the y- and z-terms are absent, then the plane is of the form x = k, and the parametric equations

$$\begin{cases} x = k \\ y = u \\ z = v \end{cases}$$

give us the plane. The u- and v-nets are horizontal and vertical lines again.

Graphs of Functions of Two Variables

$\boxed{14.1}$

The graph of a function z = f(x, y) of two variables is a surface in space. It is representable as a parametric surface in the form

$$\begin{cases} x = u \\ y = v \\ z = f(u,v) \end{cases} .$$

The u-curves of this representation are traces of the surface in planes parallel to the xz-plane, and the v-curves are traces in planes parallel to the yz-plane.

Quadric Surfaces

$\boxed{12.6}$

The various quadric surfaces can be represented as parametric surfaces using trigonometric functions. Consider the following examples.

The sphere of radius r centered at the origin can be described at each point P by specifying two angles, an angle u in the xy-plane from the positive x-axis (corresponding to θ in cylindrical coordinates) and an angle v of elevation from the xy-plane to the point P. By computing the coordinates of P, we find the parametric equations of the sphere to be

$$\begin{cases} x = r \cos u \cos v \\ y = r \sin u \cos v \\ z = r \sin v \end{cases} , u \in [0, 2\pi], v \in [-\frac{\pi}{2}, \frac{\pi}{2}].$$

The u-net consists of the circles of latitude of the sphere, and the v-net consists of the meridians, or circles of longitude.

A portion of the sphere can be obtained by restricting u or v. For example, requiring $v \in [0, \frac{\pi}{2}]$ gives us the upper hemisphere.

The ellipsoid with semi-axes a, b, and c, centered at the origin, is a slight generalization of the sphere, and has parametric equations

$$\begin{cases} x = a \cos u \cos v \\ y = b \sin u \cos v \\ z = c \sin v \end{cases}, \quad u \in [0, 2\pi], v \in [-\frac{\pi}{2}, \frac{\pi}{2}].$$

The elliptic paraboloid $z = \frac{x^2}{a^2} + \frac{y^2}{b^2}$ can be represented as for other functions of two variables if you want the u- and v-nets to be vertical plane sections. If you want horizontal sections, the parametrization

$$\begin{cases} x = au \cos v \\ y = bu \sin v \\ z = u^2 \end{cases}, \quad u \in [0, \infty), v \in [0, 2\pi]$$

has as its v-net the set of horizontal plane sections.

A hyperboloid of one sheet can be represented as

$$\begin{cases} x = a \cosh u \cos v \\ y = b \cosh u \sin v \\ z = c \sinh u \end{cases}, \quad u \in [0, \infty), v \in [0, 2\pi].$$

The u-net is the set of sections by planes containing the z-axis, and the v-net is the set of horizontal plane sections.

Most of the other quadric surfaces can be represented in similar fashion, perhaps by interchanging x, y, and z, or can be represented as functions of two variables.

Surfaces of Revolution

$\boxed{12.6}$

Suppose the curve $z = f(y)$, $x = 0$, $y \in [c, d]$ is rotated about the z-axis to obtain a surface of revolution. If u represents the angle of rotation at any point, the parametric equations of the surface are

$$\begin{cases} x = v \cos u \\ y = v \sin u \\ z = f(v) \end{cases}, \quad u \in [0, 2\pi], v \in [c, d].$$

The u-net consists of circles traced by points on the original curve, and the v-net consists of section by half-planes containing the z-axis.

Suppose the circle of radius a in the yz-plane, centered at the point (0, b, 0), is revolved about the z-axis to obtain a torus. If u is the angle of rotation, and v is the angle of elevation from the center of the rotated circle to a point on the surface, then the parametric equations of the torus are

$$\begin{cases} x = (b - a \cos v) \cos u \\ y = (b - a \cos v) \sin u \\ z = a \sin v \end{cases}, \quad u, v \in [0, 2\pi].$$

The u-net consists of circular cross-sections by half-planes containing the z-axis, and the v-net consists of horizontal cross-sections.

Other surfaces of revolution can be represented is a similar fashion.

Graphing Parametric Surfaces

Here is an environment for the graphing of parametric surfaces on the HP-28S and HP-48SX. It fits in the P.SRF subdirectory on the TH.D.G menu mentioned earlier. The complete menu under P.SRF is

XYZST	ABSTO	CDSTO	DRAGR	RCLGR	QUIT
NU	NV	U.N.V	X	Y	Z
A	B	C	D	USCR	VSCR
SCR	PPAR				

• PPAR contains the plot parameters (default parameters are best).

• SCR contains the entire plot of the surface.

• VSCR and USCR contain the screens of the v-net alone and the u-net alone.

• C and D are the limits of the v-curves; A and B are the limits of the u-curves.

• X, Y, and Z are containers for the functions that define the surface.

• U.N.V is for putting the u-net and the v-net graphs together into the display. For the HP-28S, it is the program

```
« CLLCD USCR VSCR OR DUP 'SCR' STO →LCD DGTIZ »
                                28S checksum: "F868"
```

For the HP-48SX, U.N.V is the program

```
« USCR VSCR + DUP 'SCR' STO PICT STO GRAPH »
                                48SX checksum: # 14776d
```

• NV is the number of curves in the v-net that are to be plotted. NU is the number of curves in the u-net.

• QUIT is « UP » or « UPDIR ».

• RCLGR is a subdirectory containing the three commands RCLS, RCLU, and RCLV.

•• RCLV is for recalling the v-net. For the HP-28S it is the program « VSCR →LCD UP DGTIZ » ; for the HP-48SX, it is the program « VSCR PICT STO UPDIR GRAPH ».

•• RCLU puts the graph of the u-net on the screen. For the HP-28S it is the program « USCR →LCD UP DGTIZ » ; for the HP-48SX it is the program « USCR PICT STO UPDIR GRAPH ».

•• RCLS puts the graph of the surface (both the u-net and the v-net) on the screen. For the HP-28S, it is the program « UP U.N.V »; for the HP-48SX, it is the program « UPDIR U.N.V ».

This completes the description of RCLGR.

• DRAGR is a subdirectory that contains the commands DRAS, DRAU, and DRAV, and the container PPAR.

•• PPAR contains the plot parameters (default parameters are fine).

•• DRAV sketches the v-net, and is almost the same as DRAU except that the U's and V's are interchanged; here is the program for the HP-28S:

```
« D C - NV / 'DV' STO B A - 40 / 'DU' STO A 'U' STO C 'V'
STO CLLCD 1 NV 1 + START 1 41 START X →NUM Y →NUM Z
→NUM TRANS PIXEL DU 'U' STO+ NEXT DV 'V' STO+ A 'U' STO
NEXT { U V DU DV } PURGE LCD→ UP 'VSCR' STO »
                                        28S checksum: "2476"
```

Here is the program DRAV for the HP-48SX:

```
« D C - NV / 'DV' STO B A - 40 / 'DU' STO A 'U' STO C 'V'
STO ERASE 1 NV 1 + START 1 41 START X →NUM Y →NUM Z
→NUM TRANS PIXON DU 'U' STO+ NEXT DV 'V' STO+ A 'U' STO
NEXT { U V DU DV } PURGE PICT RCL UPDIR 'VSCR' STO »
                                        48SX checksum: # 28188d
```

When the program DRAV has run on the HP-48SX, it is necessary to press GRAPH to see the picture.

•• DRAU sketches the u-net. Here is the program for the HP-28S:

```
« B A - NU / 'DU' STO D C - 40 / 'DV' STO C 'V' STO A 'U'
STO CLLCD 1 NU 1 + START 1 41 START X →NUM Y →NUM Z
→NUM TRANS PIXEL DV 'V' STO+ NEXT DU 'U' STO+ C 'V' STO
NEXT { U V DU DV } PURGE LCD→ UP 'USCR' STO »
                                        28S checksum: "C3CF"
```

Here is the program DRAU for the HP-48SX:

```
« B A - NU / 'DU' STO D C - 40 / 'DV' STO C 'V' STO A 'U'
STO ERASE 1 NU 1 + START 1 41 START X →NUM Y →NUM Z
→NUM TRANS PIXON DV 'V' STO+ NEXT DU 'U' STO+ C 'V' STO
NEXT { U V DU DV } PURGE PICT RCL UPDIR 'USCR' STO »
                                        48SX checksum: # 56812d
```

When the program DRAU has run on the HP-48SX, it is necessary to press GRAPH to see the picture.

•• DRAS draws both the u-net and the v-net and puts them together, and is the program

« DRAU DRAGR DRAV U.N.V ».

28S checksum: "64AE"
48SX checksum: # 14536d

This completes the description of DRAGR.

• CDSTO is for storing the values C and D, and is the program « 'D' STO 'C' STO ». It takes the values for C and D from the stack.

• ABSTO is just like CDSTO.

• XYZST is for storing the functions that define the surface. It is the program

« 'Z' STO 'Y' STO 'X' STO »,

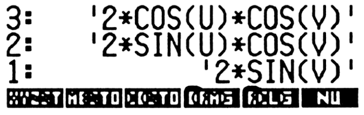
and takes the three functions from the stack. The functions should be entered in algebraic form, using U and V as the parameters.

To illustrate the use of this environment, suppose we wish to sketch the portion of the sphere of radius 2, centered at the origin, which lies in the first octant. The parametric equations for this surface are

$$\begin{cases} x = 2\cos u \cos v \\ y = 2\sin u \cos v \\ z = 2\sin v \end{cases}, \quad u \in [0, \tfrac{\pi}{2}],\ v \in [0, \tfrac{\pi}{2}].$$

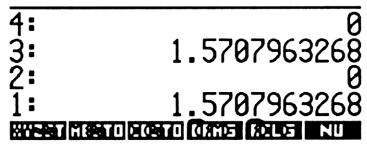
To enter the functions, we type

'2 * C O S (U) * C O S (V $\boxed{\text{ENTER}}$
'2 * S I N (U) * C O S (V $\boxed{\text{ENTER}}$
'2 * S I N (V $\boxed{\text{ENTER}}$


```
3:    '2*COS(U)*COS(V)'
2:    '2*SIN(U)*COS(V)'
1:           '2*SIN(V)'
```

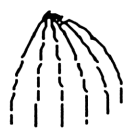
$\boxed{\text{XYZST}}$. To store the limits for u and v, since they are both the same, we type

0 $\boxed{\text{ENTER}}$ π $\boxed{\text{ENTER}}$ 2 $\boxed{\div}$ $\boxed{\to\text{NUM}}$
$\boxed{\text{ENTER}}$ 0 $\boxed{\text{SWAP}}$

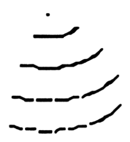

```
4:                0
3:     1.5707963268
2:                0
1:     1.5707963268
```

and then press $\boxed{\text{ABSTO}}$ and then $\boxed{\text{CDSTO}}$. Finally, we must decide how many curves of each net to draw; suppose we want 5 curves in the u-net and 4 curves in the v-net. We type 5 $\boxed{\text{ENTER}}$ ' N U $\boxed{\text{STO}}$ 4 $\boxed{\text{ENTER}}$ ' N V $\boxed{\text{STO}}$. Then we press $\boxed{\text{DRAGR}}$ and then press

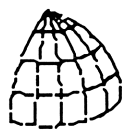
$\boxed{\text{DRAU}}$ to get the u-net only,



DRAV to get the v-net only,



or DRAS to get both nets.



Vector Functions

13.1

Vector functions, as in the context of curvilinear motion, can be handled by the HP calculators. Unfortunately, since the entries of arrays must be numbers, functions cannot be put into vectors. However, functions can be put into lists, and then vector-like routines can be written for the lists, so that the handling of vector functions can be automated.

Velocity and Acceleration

13.2

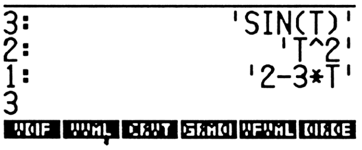
For example, the velocity and acceleration of a vector (position) function can be computed by the following little program for differentiating a vector function, called VDIF. It takes a list of three parametric functions, written in terms of the parameter T, from the stack, differentiates them, and returns both the function and its derivative to the stack.

```
« → V « V V 1 GET 'T' ∂ V 2 GET 'T' ∂ V 3 GET 'T' ∂ 3
→LIST » »
```

28S checksum: "921C"
48SX checksum: # 22898d

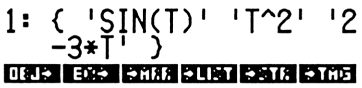
For example, given the vector function **r**(t) = [sin t, t^2, 2 - 3t], we compute the velocity by differentiating. We first construct the list for **r**, by typing

' S I N (T ENTER ' T ^ 2 ENTER
' 2 - 3 * T ENTER 3



```
3:            'SIN(T)'
2:               'T^2'
1:             '2-3*T'
3
```

→LIST (on the PRG OBJ menu).



```
1: { 'SIN(T)' 'T^2' '2
   -3*T' }
```

Then we differentiate by pressing

VDIF.

```
2: { 'SIN(T)' 'T^2' '…
1: { 'COS(T)' '2*T' -3
   }
VDIF VVAL CRVT GRADI VFVAL DIRDE
```

We calculate the acceleration by again pressing

VDIF.

```
3: { 'SIN(T)' 'T^2' '…
2: { 'COS(T)' '2*T' -…
1:   { '-SIN(T)' 2 0 }
VDIF VVAL CRVT GRADI VFVAL DIRDE
```

To evaluate a vector function at a given value of the parameter, the following function is useful. It is called VVAL. It assumes that the vector function is in level 2 and the value of T is in level 1, and it returns both the function and its value to the stack.

```
« 'T' STO DUP → V « V 1 GET →NUM V 2 GET →NUM V 3
GET →NUM 3 →LIST 'T' PURGE » »
```

28S checksum: "2F14"
48SX checksum: # 23040d

For example, given the vector function above, to evaluate it at t = 5, we create the list of parametric functions as above, and then type

5

```
1: { 'SIN(T)' 'T^2' '2
   -3*T' }
5
VDIF VVAL CRVT GRADI VFVAL DIRDE
```

VVAL.

```
2: { 'SIN(T)' 'T^2' '…
1: { -.958924274663 25
   -13 }
VDIF VVAL CRVT GRADI VFVAL DIRDE
```

Curvature

13.3

The following program, called CRVT, computes the curvature of a parametric curve in three-space, using the formula

$$\kappa = \frac{\|\mathbf{r}' \times \mathbf{r}''\|}{\|\mathbf{r}'\|^3}.$$

Because the curvature in symbolic form is usually very complicated, this program returns the value of the curvature at a point. The program can be modified, by writing routines for the cross product and magnitude of vector functions given as lists, to return the symbolic expression if preferred. This program takes a list of three parametric equations (the position function **r**) from level 2 and the value of T from level 1 and returns the value of κ.

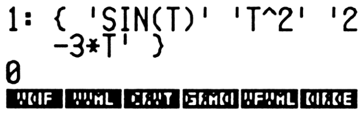
```
« → R T « R VDIF T VVAL LIST→ →ARRY 'RP' STO VDIF T
VVAL LIST→ →ARRY RP SWAP CROSS ABS RP ABS 3 ^ / 'RP'
PURGE SWAP DROP SWAP DROP » »
```

28S checksum: "FAF0"

For the HP-48SX, the command LIST→ in CRVT must be replaced by OBJ→. With that change, the 48SX checksum is # 44326d.

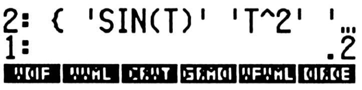
For example, to find the curvature of the curve **r**(t) = [sin t, t^2, 2 - 3t] at the point t = 0, we type

' S I N (T $\boxed{\text{ENTER}}$ ' T ^ 2 $\boxed{\text{ENTER}}$
' 2 - 3 * T $\boxed{\text{ENTER}}$ 3 $\boxed{\to\text{LIST}}$ 0



```
1: { 'SIN(T)' 'T^2' '2
   -3*T' }
0
```

$\boxed{\text{CRVT}}$.



```
2: { 'SIN(T)' 'T^2' '…
1:                  .2
```

Cylindrical Coordinates

$\boxed{12.7}$

Since the rectangular coordinates (x, y, z) and the cylindrical coordinates [r, θ, z] are related in the same way as rectangular and polar coordinates in the first two variables, the HP calculators will transform from rectangular to cylindrical coordinates. On the HP-28S, use the commands R→P and P→R; on the HP-48SX, use POLAR.

The HP-48SX will also handle spherical coordinates, as explained on p. 171 of the *HP-48SX Owner's Manual.*

Chapter 11. Partial Differentiation

The HP calculators can be used to accomplish several things in the calculus of several variables, including graphing functions of two variables, creating contour maps, computing partial derivatives, and constructing gradients and directional derivatives.

Functions of Two Variables

Graphs

$\boxed{14.1}$

The graph of the function z = f(x, y) is obtained by using the parametric graph unit of Chapter 10, with the parametric equations

$$\begin{cases} x = u \\ y = v \\ z = f(u,v) \end{cases} .$$

To get the graph of the portion of the surface above the rectangle [a, b] × [c, d], set u to run from a to b and v to run from c to d. You will get a wire mesh representation of the surface.

It is also possible to get a "hidden line" version of the graph, in which parts of the surface hidden by other parts in front of them are not shown, but it is an intensive job, perhaps best left to a larger computer.

Contour Maps

$\boxed{14.1}$

Contour maps of the function z = f(x, y) can be created on the HP-48SX by using the CONIC plot type and on the HP-28S by using the implicit graph environment IMP.G of Chapter 9. To plot the c-level curve of z = f(x, y), plot the implicit graph of f(x, y) - c = 0.

Partial Derivatives

$\boxed{14.3}$

Since the independent variable must be specified for the $\boxed{\text{d/dx}}$ command to work, partial differentiation is just the same as differentiation, but with other symbols around. The procedure for getting partial derivatives is identical to that for differentiating functions of one variable.

Programs can be developed for getting higher-order partial derivatives, mixed partials, and so forth. Creation of such programs is left to the interested reader.

Two-Variable Newton's Method

Given a system (not usually linear) of two equations in two unknowns

$$\begin{cases} f(x, y) = 0 \\ g(x, y) = 0 \end{cases},$$

there is a two-variable version of Newton's method that takes an estimate of a solution and returns (hopefully) a better estimate. Without discussing the theory, which can be found in elementary numerical analysis books, we give here a set of programs for the two-variable Newton's method. On the calculator, it behaves just like the Newton's method discussed in Chapter 4.

First comes the program FGSTO:

```
« DUP 'X' ∂ 'GX' STO DUP 'Y' ∂ 'GY' STO 'G' STO DUP 'X' ∂
'FX' STO DUP 'Y' ∂ 'FY' STO 'F' STO »
                                        28S checksum: "FF14"
                                        48SX checksum: # 23092d
```

FGSTO takes the two functions f(x, y) and g(x, y) from the stack and stores them and their partial derivatives for later use.

Then comes the program NEWT2:

```
« 'Y' STO 'X' STO X Y F GY * G FY * - →NUM FX GY * FY
GX * - →NUM DUP 'J' STO / X SWAP - FX G * F GX * -
→NUM J / Y SWAP - 'Y' STO 'X' STO X Y { X Y J } PURGE »
                                        28S checksum: "5712"
                                        48SX checksum: # 9370d
```

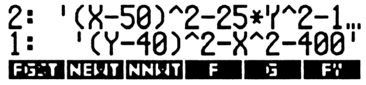
NEWT2 takes the estimate x_1 and y_1 from the stack and returns a new estimate x_2 and y_2.

As an example, to solve the system of equations

$$\begin{cases} (x - 50)^2 - 25y^2 = 100 \\ (y - 40)^2 - x^2 = 400 \end{cases},$$

we type

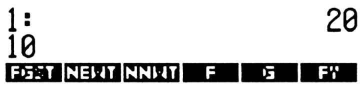
' (X - 50) ^ 2 - 25 * Y ^ 2 - 100 $\boxed{\text{ENTER}}$ ' (Y - 40) ^ 2 - X ^ 2 - 400 $\boxed{\text{ENTER}}$



```
2:  '(X-50)^2-25*Y^2-1…
1:   '(Y-40)^2-X^2-400'
```

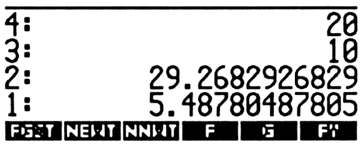
$\boxed{\text{FGSTO}}$. Then we enter an estimate; if (20, 10) is our guess, we type

20 $\boxed{\text{ENTER}}$ 10



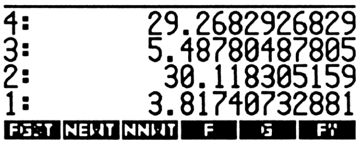
```
1:                  20
10
```

NEWT2.

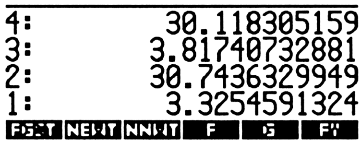


```
4:               20
3:               10
2:    29.2682926829
1:    5.48780487805
```

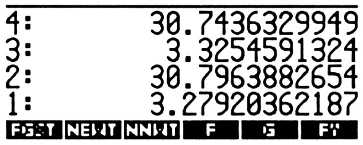
Press NEWT2 several times.



```
4:    29.2682926829
3:    5.48780487805
2:      30.118305159
1:    3.81740732881
```



```
4:      30.118305159
3:    3.81740732881
2:    30.7436329949
1:     3.3254591324
```



```
4:    30.7436329949
3:     3.3254591324
2:    30.7963882654
1:    3.27920362187
```

We can have the machine press NEWT2 for us several times, by using a program such as NNWT2:

```
« → N « 1 N START NEWT2 NEXT »
```

This program takes the estimated solution from levels 3 and 2 and the number of repetitions from level 1 and returns that many successive estimates.

These programs can be organized into a subdirectory, perhaps called NWT2. The complete menu might be

FGSTO	NEWT2	NNWT2	F	G	QUIT
FX	FY	GX	GY		

Gradients

14.5

A gradient of a function of three variables is just a vector field in which the components are the partial derivatives. Here is a simple program for computing the gradient of a function of three variables, and presenting it as a list of functions. It is called GRADI.

```
« → F « F 'X' ∂ F 'Y' ∂ F 'Z' ∂ 3 →LIST » »
                                   28S checksum: "FFC5"
                                   48SX checksum: # 50590d
```

This program takes a function of three variables, written in terms of X, Y, and Z, from the stack and returns the gradient as a list of the three partial derivatives.

For example, to find the gradient of the function F(x, y, z) = x^2y - 3z, type

'X ^ 2 * Y - 3 * Z [ENTER]

```
1:              'X^2*Y-3*Z'
 VDIF  VVAL  CRVT  GRADI VFVAL DIRDE
```

[GRADI].

```
1: { '2*X*Y' 'X^2' -3
   }
 VDIF  VVAL  CRVT  GRADI VFVAL DIRDE
```

Here is a little program, called VFVAL, for evaluating a vector field of three variables at a point. It takes the vector field, written as a list, from level 2 and the point, written as a vector, from level 1. It returns the value of the field as a vector.

```
« ARRY→ DROP 'Z' STO 'Y' STO 'X' STO → F « F 1 GET →NUM
F 2 GET →NUM F 3 GET →NUM 3 →ARRY { X Y Z } PURGE »
»
```

28S checksum: "C745"
48SX checksum: # 15924d

For example, to evaluate the vector field [2xy, x^2, 2z] at the point (1, 5, 3), type

'2 * X * Y [ENTER] 'X ^ 2 [ENTER]
'2 * Z [ENTER] 3 [→LIST]

```
1: { '2*X*Y' 'X^2' '2*
   Z' }
 OBJ→  EQ→  →ARR  →LIST →STR  →TAG
```

[1 , 5 , 3 [ENTER]

```
2: { '2*X*Y' 'X^2' '2…
1:             [ 1 5 3 ]
 VDIF  VVAL  CRVT  GRADI VFVAL DIRDE
```

[VFVAL].

```
1:            [ 10 1 6 ]
 VDIF  VVAL  CRVT  GRADI VFVAL DIRDE
```

Directional Derivatives

[14.5]

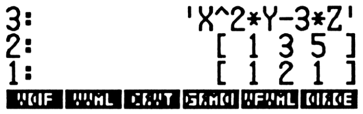
Vector-handling routines on the HP calculators make the computation of the directional derivative easy to program. To compute the directional derivative of a function f at a point P in the direction of a vector **v**, we form the dot product of the gradient of f at P with a unit vector in the direction of **v**. The following program, called DIRDE, takes the function f from level 3, the point P (written as a vector) from level 2, and the vector **v** from level 1, and returns the directional derivative.

```
« → F P V « F GRADI P VFVAL V V ABS / DOT » »
```

28S checksum: "E3A5"
48SX checksum: # 31755d

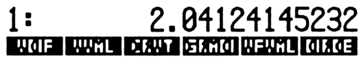
For example, to find the directional derivative of the function f(x, y, z) = x^2y - 3z at the point (1, 3, 5) in the direction of **v** = [1, 2, 1], type

'X ^ 2 * Y - 3 * Z $\boxed{\text{ENTER}}$ [1 , 3 , 5 $\boxed{\text{ENTER}}$ [1 , 2 , 1 $\boxed{\text{ENTER}}$



```
3:           'X^2*Y-3*Z'
2:             [ 1 3 5 ]
1:             [ 1 2 1 ]
```

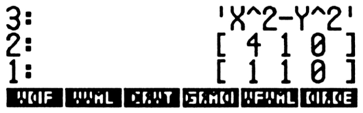
$\boxed{\text{DIRDE}}$.



```
1:        2.04124145232
```

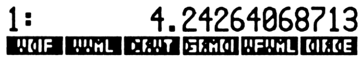
This same program will also work for functions of two variables; just put in zero for the third component. For example, to find the directional derivative of f(x, y) = $x^2 - y^2$ at the point (4, 1) in the direction of v = [1, 1], type

'X ^ 2 - Y ^ 2 $\boxed{\text{ENTER}}$ [4 , 1 , 0 $\boxed{\text{ENTER}}$ [1 , 1 , 0 $\boxed{\text{ENTER}}$



```
3:             'X^2-Y^2'
2:             [ 4 1 0 ]
1:             [ 1 1 0 ]
```

$\boxed{\text{DIRDE}}$.



```
1:        4.24264068713
```

Chapter 12. Path Integrals and Multiple Integrals

Capabilities of the HP symbol-manipulating calculators enable the evaluation of path integrals and multiple integrals in a very efficient manner. Here are some programs that do these things, taking much of the drudgery out of applications.

Path Integrals

$\boxed{15.3}$

The path integral of the vector field **F** = ⟨ M(x, y, z), N(x, y, z), P(x, y, z) ⟩ along the curve C: **r**(t) = ⟨ x(t), y(t), z(t) ⟩, t ∈ [a, b], is $\int_C \mathbf{F}\cdot d\mathbf{r} = \int_a^b \mathbf{F}(\mathbf{r}(t))\cdot\mathbf{r}'(t)\,dt$. An environment for the evaluation of such path integrals is presented here.

Start with a subdirectory for path integrals, perhaps called P.INT, which contains the following programs and containers:

FSTO	RSTO	ABSTO	PINT	IERR	QUIT
M	N	P	X	Y	Z
A	B	XP	YP	ZP	

- XP, YP, and ZP are storage containers for the components of the derivative **r**'.
- A and B are storage containers for a and b.
- X, Y, and Z are storage containers for the components of **r**.
- M, N, and P are storage containers for the components of **F**.
- QUIT is just the program « UP » or « UPDIR ».
- IERR is a container for the error of integration.
- PINT is the program that does the calculation. For the HP-28S it is

```
« M EVAL XP * N EVAL YP * + P EVAL ZP * + { T A B }
.00001 ∫ 'IERR' STO »
                                        28S checksum: "DCF3"
```

For the HP-48SX the program PINT is

```
« RCLF A B M EVAL XP * N EVAL YP * + P EVAL ZP * + 'T'
5 FIX ∫ →NUM SWAP STOF »
                                        48SX checksum: # 15922d
```

This program evaluates **F** at **r**(t), forms the dot product with **r**'(t), and then uses the machine's built-in numerical integration routine to evaluate the integral. The number .00001 is specified as the error tolerance. The result of the program is two numbers, the value of the integral and the maximum error.

• ABSTO is the program « 'B' STO 'A' STO » for storing the limits a and b of the parameter, describing the curve C. These become the limits of the integral that is evaluated.

• RSTO is the program

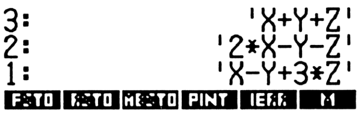
« DUP 'T' ∂ 'ZP' STO 'Z' STO DUP 'T' ∂ 'YP' STO 'Y' STO DUP 'T' ∂ 'XP' STO 'X' STO »

for storing the vector function **r** of which the curve C is the graph. The program also computes and stores the derivative **r**'. The components X, Y, and Z of **r** should be entered onto the stack in that order, using T as the parameter.

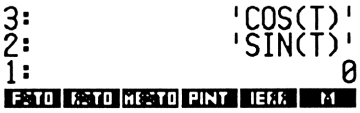
• FSTO is the program « 'P' STO 'N' STO 'M' STO » for storing the three components of the vector field **F**. The components M, N, and P should be entered onto the stack in that order, using variables X, Y, and Z.

To illustrate, suppose we wish to find the path integral of **F**(x, y, z) = ⟨ x + y + z, 2x - y - z, x - y + 3z ⟩ over the circle C: **r**(t) = ⟨cos t, sin t, 0 ⟩, t ∈ [0, 2π]. We type

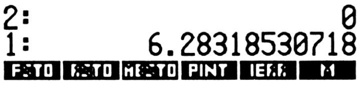
' X + Y + Z $\boxed{\text{ENTER}}$ ' 2 * X - Y - Z $\boxed{\text{ENTER}}$ ' X - Y + 3 * Z $\boxed{\text{ENTER}}$


```
3:          'X+Y+Z'
2:        '2*X-Y-Z'
1:        'X-Y+3*Z'
```

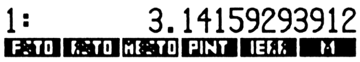
$\boxed{\text{FSTO}}$
' C O S (T $\boxed{\text{ENTER}}$ ' S I N (T $\boxed{\text{ENTER}}$ 0 $\boxed{\text{ENTER}}$


```
3:         'COS(T)'
2:         'SIN(T)'
1:                0
```

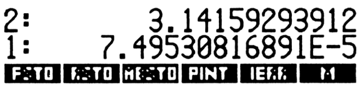
$\boxed{\text{RSTO}}$
0 $\boxed{\text{ENTER}}$ π $\boxed{\text{ENTER}}$ 2 * $\boxed{\text{→NUM}}$


```
2:                0
1:    6.28318530718
```

$\boxed{\text{ABSTO}}$ $\boxed{\text{PINT}}$.


```
1:    3.14159293912
```

Press $\boxed{\text{IERR}}$ to see the possible error.


```
2:    3.14159293912
1: 7.49530816891E-5
```

Numerical Multiple Integration

16.1,16.4

The most straightforward approach to numerical integration for multiple integrals is to express them as iterated integrals and then use a nested Simpson's rule.

The version of Simpson's rule developed in Chapter 7 has quite a bit of inefficiency built into it, so we will first give a streamlined version of Simpson's rule for a definite (simple) integral, then for double iterated integrals, and finally for triple iterated integrals.

It is best to organize these programs into three subdirectories of a single directory M.INT (for Multiple INTegration). The three subdirectories could be called INT1, INT2, and

INT3, or maybe S.INT, D.INT, and T.INT, for simple, double, and triple integrals, respectively. We will choose the former.

Simple Integrals

$\boxed{7.8}$

In the subdirectory INT1, there are several programs and storage containers for evaluating integrals of the form

$$\int_a^b f(x)\,dx.$$

The complete menu under INT1 is:

FABST	NSTO	SIMP1	F	N	QUIT
A	B	H			

• H is a container for storing the step size.

• A and B are containers for storing the limits of the integral.

• QUIT is just the program « UP » or « UPDIR », as you probably guessed.

• N is a container for storing half the number of subintervals. This number N is the same as the number N in Chapter 7; in Simpson's' rule, the number of subintervals is 2N, since midpoints of the major subintervals are used, too.

• F is a container for storing the integrand f(x).

• SMP1 is the fast version of Simpson's rule for the integral, and is the program

```
« A 'X' STO F EVAL 1 N 2 * 1 - FOR I H X + 'X' STO F
EVAL 2 * DUP 'T' STO + I 2 / DUP IP IF ≠ THEN T + END
NEXT B 'X' STO F EVAL + H * 3 / { X T } PURGE »
```

28S checksum: "FC4E"
48SX checksum: # 22096d

• NSTO is the program

```
« 'N' STO B A - N 2 * / 'H' STO »,
```

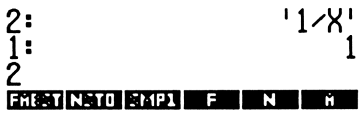
28S checksum: "9CBE"
48SX checksum: # 48933d

which not only stores N, half the number of subintervals involved, but also computes H, the step size.

• FABST is the program « 'B' STO 'A' STO 'F' STO » for storing the integrand f(x) and the limits a and b. The name of the program reminds you to enter f, then a, and finally b. The function f should be written in algebraic form in terms of the variable X.

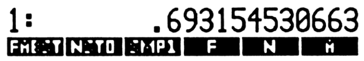
To illustrate the use of this environment, suppose we wish to evaluate the integral
$\int_1^2 \frac{1}{x}\,dx$, using N = 4. We would type the following:

' 1 / X $\boxed{\text{ENTER}}$ 1 $\boxed{\text{ENTER}}$ 2



```
2:              '1/X'
1:                  1
2
```

$\boxed{\text{FABST}}$ 4 $\boxed{\text{NSTO}}$ $\boxed{\text{SMP1}}$.



```
1:        .693154530663
```

Double Integrals

$\boxed{16.1}$

The subdirectory INT2 consists of programs and containers for evaluating double integrals of the form

$$\int_a^b \int_{c(x)}^{d(x)} f(x, y)\,dy\,dx.$$

The complete menu under INT2 is

FABST	CDSTO	MNST	SMP2	F	QUIT
M	N	A	B	CX	DX
KSTO	SMP1	H			

• H is a container for storing the "outer" stepsize.

• SMP1 is the program

```
« C 'Y' STO F EVAL 1 N 2 * 1 - FOR J K Y + 'Y' STO F
EVAL 2 * DUP 'T' STO + J 2 / DUP IP IF ≠ THEN T + END
NEXT D 'Y' STO F EVAL + { Y T } PURGE »
```

28S checksum: "E714"
48SX checksum: # 60313d

• KSTO is the program

```
« 'X' STO DX EVAL DUP 'D' STO CX EVAL DUP 'C' STO - N 2 *
/ 'K' STO »
```

28S checksum: "EF"
48SX checksum: # 31931d

• A, B, CX, and DX are containers for storing the limits a, b, c(x), and d(x).

• M and N are containers for storing M and N, half the numbers of intervals in the inner and outer subdivisions.

• QUIT is just « UP » or « UPDIR ».

• F is a container for storing the integrand f(x, y).

• SMP2 is the main program:

```
« A KSTO SMP1 1 M 2 * 1 - FOR I H X + KSTO SMP1 2 *
DUP 'T' STO + I 2 / DUP IP IF ≠ THEN T + END NEXT B KSTO
SMP1 + H * 3 / { X T C D K } PURGE »
```

28S checksum: "DEA6"
48SX checksum: # 2151d

• MNST is the program

```
« 'N' STO 'M' STO B A - M 2 * / 'H' STO »,
```

28S checksum: "D6A1"
48SX checksum: # 56443d

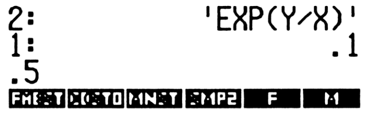
for storing the numbers M and N, there being 2M subintervals in the X-direction and 2N subintervals in the Y-direction. It also computes the step size H in the X-direction.

• CDSTO is the program « 'DX' STO 'CX' STO », for storing the limits c(x) and d(x). They are to be entered in algebraic form in terms of the variable X, and in that order.

• FABST is the program « 'B' STO 'A' STO 'F' STO », for storing the integrand and the first two limits. The function f(x, y) should be entered in algebraic form, using the variables X and Y.

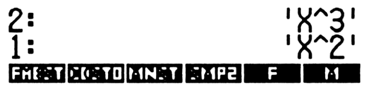
As an illustration of the use of this environment, suppose that we wish to estimate the iterated integral $\int_{0.1}^{0.5}\int_{x^3}^{x^2} e^{y/x}\,dy\,dx$ with M = N = 5. We would type the following:

'E X P (Y / X [ENTER] .1 [ENTER] .5



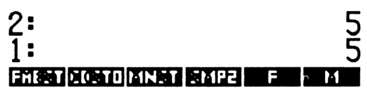
```
2:          'EXP(Y/X)'
1:                  .1
.5
```

[FABST]
' X ^ 3 [ENTER] ' X ^ 2 [ENTER]



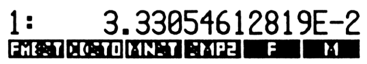
```
2:               'X^3'
1:               'X^2'
```

[CDSTO]
5 [ENTER] [ENTER]



```
2:                   5
1:                   5
```

[MNST] [SMP2].



```
1:    3.33054612819E-2
```

Triple Integrals

$\boxed{16.4}$

The subdirectory INT3 contains programs and containers for evaluating triple integrals of the form

$$\int_a^b \int_{c(x)}^{d(x)} \int_{e(x,y)}^{g(x,y)} f(x, y, z)\, dz\, dy\, dx.$$

The complete menu under INT3 is as follows:

FABST	CDEGS	MNPS	SMP3	F	QUIT
M	N	P	A	B	CX
DX	EXY	GXY	KSTO	SMP2	LSTO
SMP1	H				

• H is a container for storing the stepsize H.

• SMP1 is the program

```
« E 'Z' STO F EVAL 1 P 2 * 1 - FOR U L Z + 'Z' STO F EVAL
2 * DUP 'T' STO + U 2 / DUP IP IF ≠ THEN T + END NEXT G
'Z' STO F EVAL + L * 3 / { Z T } PURGE »
```

28S checksum: "4F6B"
48SX checksum: # 31372d

• LSTO is the program

```
« 'Y' STO GXY EVAL DUP 'G' STO EXY EVAL DUP 'E' STO - P 2
* / 'L' STO »
```

28S checksum: "E9FF"
48SX checksum: # 38671d

• SMP2 is the program

```
« C LSTO SMP1 1 N 2 * 1 - FOR J K Y + LSTO SMP1 2 * DUP
'T' STO + J 2 / DUP IP IF ≠ THEN T + END NEXT D LSTO
SMP1 + K * 3 / { Y T } PURGE »
```

28S checksum: "A485"
48SX checksum: # 8655d

• KSTO is the program

```
« 'X' STO DX EVAL DUP 'D' STO CX EVAL DUP 'C' STO - N 2 *
/ 'K' STO »
```

28S checksum: "EF"
48SX checksum: # 31931d

• A, B, CX, DX, EXY, and GXY are containers for storing the limits of the integrals.

• M, N, and P are containers for storing the numbers M, N, and P.

• QUIT is « UP » or « UPDIR ».

• F is a container for storing the integrand f(x, y, z).

• SMP3 is the main program:

```
« A KSTO SMP2 1 M 2 * 1 - FOR I H X + KSTO SMP2 2 *
DUP 'T' STO + I 2 / DUP IP IF ≠ THEN T + END NEXT B KSTO
SMP2 + H * 3 / { X T C D E G K L } PURGE »
                                        28S checksum: "CE65"
                                        48SX checksum: # 18400d
```

• MNPS is the program

```
     « 'P' STO 'N' STO 'M' STO B A - M 2 * / 'H' STO »
                                        28S checksum: "6F0E"
                                        48SX checksum: # 40817d
```

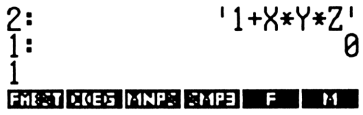
for storing the numbers M, N, and P that determine the numbers of subintervals in each direction, and also computing the outermost stepsize H.

• CDEGS is the program « 'GXY' STO 'EXY' STO 'DX' STO 'CX' STO » for storing the limits c(x), d(x), e(x, y), and g(x, y). These functions should be entered in that order, using X and Y as the variables.

• FABST is the program « 'B' STO 'A' STO 'F' STO », for storing the integrand f(x, y, z) and the limits a and b. The function f should be entered in algebraic form, using the variables X, Y, and Z.

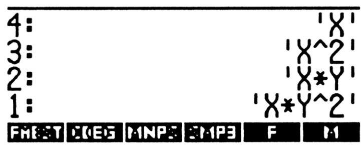
As an illustration in the use of this environment, suppose we wish to evaluate the integral $\int_0^1 \int_x^{x^2} \int_{xy}^{xy^2} (1 + xyz)\,dz\,dy\,dx$, using M = N = P = 4. We would type

' 1 + X * Y * Z $\boxed{\text{ENTER}}$ 0 $\boxed{\text{ENTER}}$ 1



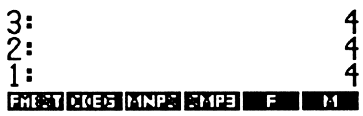
```
2:            '1+X*Y*Z'
1:                    0
1
```

$\boxed{\text{FABST}}$
' X $\boxed{\text{ENTER}}$ ' X ^ 2 $\boxed{\text{ENTER}}$ ' X * Y $\boxed{\text{ENTER}}$ ' X * Y ^ 2 $\boxed{\text{ENTER}}$



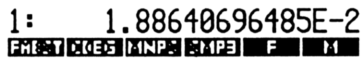
```
4:                  'X'
3:                'X^2'
2:                'X*Y'
1:              'X*Y^2'
```

$\boxed{\text{CDEGS}}$
4 $\boxed{\text{ENTER}}$ $\boxed{\text{ENTER}}$ $\boxed{\text{ENTER}}$



```
3:                    4
2:                    4
1:                    4
```

$\boxed{\text{MNPS}}$ $\boxed{\text{SMP3}}$.



```
1:     1.88640696485E-2
```

Applications of multiple integrals proceed just as in the examples above. In every case, construct the multiple integral, express it as an iterated integral, and use the appropriate program to evaluate it.

Index

ISBN 0-02-340599-6

90000

9 780023 405990