Step-by-Step Solutions
For Your HP Calculator

Technical Applications

HP-27S
HP-19B

HEWLETT
PACKARD

Technical Applications

Step-by-Step Solutions for Your HP-27S or HP-19B Calculator

HEWLETT PACKARD

Edition 2 November 1988
Reorder Number 00027-90044

Notice

This book and any keystroke programs contained herein are provided "as is" and are subject to change without notice. **Hewlett-Packard Company makes no warranty of any kind with regard to this book or the keystroke programs contained herein, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose.** Hewlett-Packard Company shall not be liable for any errors or for incidental or consequential damages in connection with the furnishing, performance, or use of this book or the keystroke programs contained herein.

Corvallis Division
1000 N.E. Circle Blvd.
Corvallis, OR 97330, U.S.A.

Printing History

Edition 1	November 1987	Mfg. No. 00027-90045
Edition 2	November 1988	Mfg. No. 00027-90053

Contents

How To Use This Book

By purchasing this book, *Technical Applications*, you've shown that you're interested in getting the most from your Hewlett-Packard calculator. In the pages that follow, we'll help you achieve that goal.

This book divided into two chapters. In Chapter 1, advanced techniques for writing Solver equations are discussed. In Chapter 2, a number of science and engineering-related applications and examples are presented. It is *not* necessary to read the material on advanced Solver equation-writing before using an application. Simply turn to the appropriate topic in Chapter 2 and enter the equation you need. Likewise, it is *not* necessary to understand the specific applications if you only wish to learn the advanced Solver techniques described in Chapter 1. How you use the book is best determined by your particular needs and wants. However, before you try the examples in this book, you should be familiar with certain concepts from the owner's manual:

- The basics of your calculator – how to write Solver equations and assign values to variables, how to solve for an unknown variable in an equation, how to recall Solver variables, how to move from menu to menu, and how to enter and edit lists.
- Entering numbers and using the built-in functions (LN, %, TAN, ABS, etc.)

This portion of the book contains important information on the examples in this book and provides a quick review of selected calculator operations. While this book presents basic equations and theory for each application topic, it should *not* be considered a text on these subjects. For more information on the topics encountered, you can consult the references listed at the end of each topic or any good textbook.

The examples in this book demonstrate approaches to solving certain problems, but by no means exhaust the many possible ways to obtain an answer.

Please take a moment to familiarize yourself with the formats used in this book.

Keys and Menu Selection

A box represents a key on the calculator keyboard:

[EXIT]
[=]
[STO]
[+]
[INPUT]

The shift key is represented by the symbol ■. Thus, shifted keys appear as:

■ [LAST]
■ [CLEAR DATA]
■ [MODES]
■ [SHOW]

A menu label is represented like this:

[PMT]	(found in the [TVM] menu)
[STDEV]	(found in the [STAT] menu)
[DOT]	(a user-created variable in a Solver equation)

Some menus contain sub-menus accessed by pressing the appropriate menu key (these are listed in the owner's manual). Also, some menus include more labels than can be displayed above the six redefinable menu keys. Press [MORE] to see the other menu options.

Display Formats and Numeric Input

The examples in this book will use a display format of 4 decimal places (FIX 4) except where noted. If your display is set such that the numeric displays do not match exactly, you can modify your display format with the [MODES] menu on the HP-27S or the [DISP] menu on the HP-19B, and the [FIX] key within these menus. If you wish to see the full 12-digit precision of a number regardless of the display format, simply press ■ [SHOW]; the full precision number is displayed as long as you hold down the [SHOW] key.

Negative numbers are created using the [+/-] key:

Keys:	**Display:**
39.087 [+/-] [=]	-39.0870
2.9 ■ [E] 30 [+/-] [=]	-2.9000E30

Negative exponents of ten are created using the [-] key:

Keys:	**Display:**
1.408 ■ [E] [-] 27 [=]	1.4080E-27
2.55 ■ [E] [-] 15 [+/-] [=]	-2.5500E-15

Entering Equations

When entering equations on your calculator, follow the instructions in the Solver chapter of your owner's manual, and in "Advanced Solver Techniques." Here are hints to help you in common error situations:

- If the calculator displays `INVALID EQUATION` when you press [CALC], the calculator does not understand something in the equation. When the equation returns to the display, the cursor blinks where the calculator detected the error. Check the equation in the display against the equation in the book. Make sure the parentheses match and the operators are where they should be. If the equation includes a logical operator ("AND" or "OR"), make certain that you include a space on both sides of the operator.
- If the calculator accepts the equation but your answer does not match the example, check the values stored in the variables by recalling them (press [RCL] then the menu key). If the values are correct, return to the [SOLVE] menu by pressing [EXIT] and check the equation against the one in this book for accuracy. When you find an error, edit the equation and then press [CALC] to display the menu of variables again.
- If the calculator displays `INSUFFICIENT MEMORY` when you press [INPUT] or [CALC], you must clear portions of memory before continuing. Refer to "Managing Calculator Memory" in appendix A of your owner's manual and to the conclusion of "Advanced Solver Techniques" in this book for additional information.

The equations in this book use variable names that are intended to remind you of what to store. Feel free to change them to something more meaningful to you.

Our thanks to Steven J. Sabin of Oregon State University for developing the problems and Solver equations in this book.

1

Advanced Solver Techniques

To help you utilize the full power and flexibility of your calculator, several advanced techniques and two new and powerful functions, LET and GET, are introduced in this chapter. As was mentioned in "How To Use This Book," it is *not* necessary to read this chapter if you are only interested in using a specific application. Simply turn to the appropriate topic in Chapter 2 and key in the equation. However, the Solver equations used in the applications incorporate many of the advanced techniques presented here. Thus, if you want to *understand* what you are keying in, read this chapter first.

Using LET and GET

Function Descriptions

These two functions are not covered in your owner's manual. However, you'll find them useful in a variety of applications. LET assigns the value of an algebraic expression (or number) to a specific variable. GET recalls the contents of a specific variable. The format for these two functions is:

Function:	**Description:**
L (*variable name* : *algebraic expression*)	Evaluates the algebraic expression, stores the result in the specified variable, and also returns that result as the value of the LET function.
G (*variable name*)	Returns the contents of the specified variable.

Like Σ and IF, LET and GET are for use only in Solver equations. Thus, you will not find these functions on a calculator key or in any menu.* To use LET and GET in a Solver equation, simply type the letters G or L and include the parentheses around the arguments. If a variable appears *only* as the first argument of a LET function and/or *only* as the argument of a GET function, it will *not* appear in the menu of variables.

There are many ways in which LET and GET enhance the capabilities of your Solver, and we will describe each of these in the pages that follow. First, a few examples will help introduce you to these two powerful functions.

* The GET here has no relation to GET in the [TEXT], [SUM], and [CFLO] menus of the HP-19B, or to GET in the [STAT] menu of the HP-27S.

Example 1. The Solver equation `A=L(D:B+C)+SIN(D)` stores the sum of B and C in D, and adds this result to the sine of D when calculating A. The LET function causes the value of D used in the argument of the sine function to be $B + C$. Notice that when this equation is "CALC"ed, A, B, C, and D all appear in the menu, but it is not necessary to explicitly enter a value for D since the LET function does so automatically.

Example 2. The Solver equation `A=L(D:B+C)+SIN(G(D))` is functionally identical to the last example when calculating a result for A, but here the GET function is used in computing the sine of D. When this equation is "CALC"ed, D does *not* appear in the menu. This is because D only appears as the first argument of the LET function and as the argument of the GET function in this Solver equation. In the last example, D was the argument of the SIN function (an appearance outside the first argument of LET or GET), and thus *did* appear in the menu.

Intermediate Variables

Normally, whenever you use a variable in a Solver equation, it will appear in the menu. When variables are used in this manner they are referred to as *formal variables* or simply *variables* when no distinction is necessary. However, there are two cases in which a variable will *not* appear in the menu:

1. The only occurrence of a variable is as the *counter variable* of the Σ function.
2. The only occurrence of a variable is as the first argument of LET and/or as the argument of GET.

The second case was illustrated in Example 2 for the variable D. When a variable is used *only* as the first argument of LET *and* as the argument of GET it is given the special name *intermediate variable*. This is because intermediate variables hold intermediate results that can be used repeatedly in an equation, even though such variables do not appear in the menu. The user cannot directly assign a value to an intermediate variable. Keeping an intermediate variable from appearing in the menu avoids confusion since the menu prompts only for relevant variables.

Example 3. The equation $a = b + c - \ln(b + c) + x^{b+c}$ can be implemented by the Solver equation

```
A=L(D:B+C)-LN(G(D))  +X^G(D).
```

Here, D serves as an intermediate variable and will not show up in the menu. The use of D in this manner avoids having to type `B+C` more than once in the equation and keeps D from appearing in the menu, a source of possible confusion since there is no variable d in the original equation.

Reducing Keystrokes With LET

In some instances you may wish to use the LET function to reduce keystrokes (as shown in the previous examples), yet still view the value of the variable used with the LET function. This is accomplished by simply including the variable *formally* one or more times in your equation, thus causing it to appear in the menu of variables. An example will help clarify this.

Example 4. The Solver equation from Example 3 can be changed slightly to `A=L(D:B+C)-LN(D)+X^D`. Notice that the GET function has simply been removed and now D appears formally two times; as the argument of ln, and as the power to which x is raised. We have still employed LET to reduce redundant keystrokes (typing `B+C` more than once), but now the sum $b + c$ can be viewed by recalling D.

A practical application of these last two techniques is in the calculation of planetary orbits as shown in the next example.

Example 5: Orbits of Planets. The equation in polar coordinates for a planet's orbit about the sun is given by:

$$r = \frac{pe}{1 + e\cos\theta}$$

where:

$$e = \frac{r_0 v_0^2}{GM} - 1.$$

Here, r is the orbital radius, θ is the angle swept out by the planet as it orbits, p is the distance between the focus and directrix (the orbit is always one of the conic sections), r_0 is the planet's orbital radius at its closest approach to the sun, v_0 is its speed at the point of closest approach, G is the universal gravitational constant, and M is the mass of the sun.

To simplify the orbital equation, we have used the variable e. You are probably familiar with this method of notation for complicated expressions. In an analogous manner, repetitive keystrokes can be eliminated in a Solver equation by using the variable E. The following Solver equation for the planet's orbit uses `ANG` to represent θ.

```
R=P×L(E:R0×SQ(V0)÷(G×M)-1)÷(1+E×COS(ANG))
```

Here we have used the LET function to assign a value to E and thus eliminate the need to type the rather long expression for E again later in the equation. Since the variable e in the original equations has special significance (the eccentricity of orbit), E is not used as an intermediate variable; instead, it appears formally as a multiplier of $\cos\theta$ and thus appears in the menu of variables. If it were not necessary to view E, the Solver equation

```
R=P×L(E:R0×SQ(V0)÷(G×M)-1)÷(1+G(E)×COS(ANG))
```

could be used. Notice that the only change made from the previous Solver equation is to employ the GET function with E. We have still used LET to reduce keystrokes by assigning a value to E, and the equation is functionally identical to the previous Solver equation when R is solved for. However, now E does *not* appear in the menu. In this case, E is used as an intermediate variable.

How LET and GET Change an Equation

In many cases, when an unknown variable appears only once as a formal variable, the Solver algebraically rearranges the equation to isolate a *direct solution*. However, if an unknown variable appears formally more than once, the direct solution method *always* fails and the Solver attempts to locate a solution iteratively. When a variable appears as the first argument of LET or as the argument of GET, it is *not* considered by the Solver in determining whether a direct solution can be found. Thus, a variable may occur many times in an equation, yet only once formally. In these instances, a direct solution may be found, but it will not normally be "correct" mathematically. The next example will clarify this.

Example 6. The Solver equation `A=2×A+B` is solved iteratively for A (since A appears twice formally) such that $A = -B$. On the other hand, the Solver equation `A=2×G(A)+B` is treated much differently. When A is solved for, a direct solution is found (A appears formally only once). The Solver multiplies the current contents of A by 2, adds the contents of B, and stores this result as the new value of A.

Using New and Old Values

As shown in "How LET and GET Change and Equation," the Solver will often return a solution that is not "correct" in the strict mathematical sense when LET and GET occur in an equation. Actually, this result is quite useful and allows LET and GET to be used to assign new values to variables using the values they currently contain. This technique can be used in recursive problems; i.e., problems in which the next value of the output is dependent on the old output.

Whenever you encounter an equation in which the unknown variable appears as both a formal variable and as the argument of a GET function, the GET function will use the current value of the variable (the value of the variable when the calculation is initiated).

A helpful aid in understanding this is to think of the variable as behaving in two different ways:

1. Where it appears formally, the variable is used as it would normally be in finding a solution to an equation; i.e., it can be algebraically rearranged to find a direct solution or used to find an iterative solution.
2. As the argument of a GET function, the variable is treated as though it were a constant, not an unknown. The value used is the current value as previously described.

In the LET function, the Solver does not check to see if a variable appears on both sides of the colon. Instead, it simply evaluates the algebraic expression on the right side of the colon using the current values of all variables. This result is assigned as the new value of the variable on the left side of the colon.

Some specific examples will demonstrate these features.

Example 7. Consider the Solver equation `A=G(A)+1`. Since A only appears once formally, the direct Solver is used. This equation recalls the *current* value of A (by the GET function) then adds 1. This result is then assigned as the *new* value of A. Thus, this simple equation increments A by 1 each time A is calculated.

Example 8. Like the last example, the Solver equation `Q=L(A:A+1)` also increments A by 1. In this case, Q is calculated instead of A. As mentioned above, the right side of the colon in the LET argument (`A+1`) is evaluated using the current value of A and this result is assigned as the new value of A. Note that A appears on the right side of the colon in the LET argument and therefore appears in the menu when the equation is "CALC"ed.

The next example shows a practical use of these techniques.

Example 9: Taylor Series Expansion. The Taylor Series expansion for e is

$$e = \sum_{j=0}^{\infty} \frac{1}{j!}.$$

How many terms are needed in the series to express e accurately to 4 decimal places?

The Solver equation `E=G(E)+1÷FACT(L(J:J+1))` will accomplish this. The GET used with E causes E to appear only once formally, and thus the direct method is used when E is solved for. The GET technique of Example 7 is used to add a new term $(1/j!)$ to the current value of E each time E is calculated. This new result is then returned as the "solution" for E.

Using the LET technique of Example 8, the equation increments J by 1 each time E is calculated. When the equation is "CALC"ed, E and J appear in the menu of variables. You should set E initially by storing a zero in it, since the sum should begin by GETting a zero value for E. From the defining equation for the series expansion of e, the sum should start at $j=0$. J in the Solver equation is set to -1 initially so that the value of J *used by the factorial function* is zero the first time E is calculated.

To view the full precision of the number as the Solver adds each term in the series, select the display format "ALL". You will find that E must be calculated 8 times to express e accurately to 4 decimal places. Thus, 8 terms ($j=0$ to 7) are needed to achieve the specified accuracy.

When Not to Use LET and GET

There are times when you *must* iterate to find the solution to an equation. The previous examples have shown how LET and GET *change* the way an equation is evaluated when their arguments are unknowns. Sometimes this can result in a *direct* solution that is not the *desired* solution.

Consider the equation $x = -e^x$. This is a *transcendental* equation; i.e., it cannot be algebraically manipulated to isolate a solution for x. Here it is *necessary* to iteratively find a solution. If you write the Solver equation `X=-EXP(G(X))` with hopes that the Solver will somehow find a direct solution (since, formally, X only appears once), you are asking for a mathematical impossibility. The Solver equation above takes the base e, raises it to the Xth power based on an initial value of X, and assigns the negative of this result as the final value of X.

Arranging Menu Variables

At times you may wish to arrange your equation variables in a specific order in the menu. There are two ways to do this.

LET and GET Method. Recall that a variable only appears in a menu when it appears *formally* in an equation. When you press ≣ CALC ≣ after entering an equation, the Solver scans from left to right in your equation and assigns variables to the menu in the order in which they are encountered. A variable is *not* considered by the Solver in menu assignments under the following conditions:

1. When a variable is used as a *counter variable* in the Σ function.
2. When a variable is used as the first argument of LET.
3. When a variable is used as the argument of GET.

This technique is shown in the example below.

Example 10. The equation $a = ln(bd) + e^c + d^2$ can have its variables arranged alphabetically using the Solver equation `A=LN(B×G(D))+EXP(C)+SQ(D)`. Since D first occurs as the argument of a GET function, it is not assigned to the menu until it is encountered as a formal variable in the term `SQ(D)`.

Note

This technique of arranging menu variables is most useful when certain variables will *not* be unknowns. If the Solver equation of Example 10 is used to calculate D, the GET function causes new and old values of D to be used rather than a true algebraic solution. If the equation always calculates variables other than D, it is perfectly acceptable.

Multiplication by 0 Method. Unlike the LET and GET method, multiplication by zero does *not* cause the Solver to use new and old values of a variable. Instead, the true mathematical integrity of the equation is preserved. The next example shows how.

Example 11. The Solver equation of Example 10 can be rewritten as `A=LN((B+0×C)×D)+EXP(C)+SQ(D)`. This will arrange the menu variables alphabetically and still allow *all* variables to be calculated. Notice that as the Solver scans this equation, it encounters the formal variables in the order A, B, C, and D. The multiplication by zero adds nothing to the argument of the natural logarithm function and its sole purpose is to arrange the variables alphabetically in the menu.

Note

Multiplication by zero causes variables that appear only once in a defining equation to appear more than once in a Solver equation. If these variables are unknown, the Solver will iteratively locate a solution since they appear formally more than once. For example, C appears formally *twice* in the Solver equation above even though it appears only once in the defining equation of Example 10. Thus, when C is calculated, the Solver will locate a solution iteratively.

Solving for More Than One Variable at a Time

In the last example, multiplication by zero was used to arrange the menu of variables in a Solver equation. Multiplication by zero has another very powerful use in Solver equations...assigning a result to *more than one* variable when a single unknown is calculated. This is done by employing the LET function and multiplication by zero. An example will illustrate this technique.

Example 12: Nautical Depth Conversions. Nautical depths are often measured in *fathoms*. Since most people do not "think" in terms of fathoms, conversion to more customary units is desirable. To convert from fathoms to feet, multiply by 6.000012; to convert from fathoms to meters multiply by 1.828804. A Solver equation that converts fathoms to feet *and* meters at the same time is

```
FT=FATH×6.000012+0×L(M:FATH×1.828804)+0×M.
```

Here we assume that fathoms (*FATH*) is the known quantity, and feet (*FT*) and meters (*M*) are unknown. Notice that when *FT* is calculated, the Solver multiplies *FATH* by the proper conversion factor. Then, the LET function *is multiplied by zero* so that its value will not affect the value of *FT*. The LET function will assign the proper value to *M*. We have also included the term `0×M` to cause *M* to appear formally and hence, in the menu of variables. When *FT* is calculated, the Solver returns a proper result *and* stores the number of meters in *M*. To see *M*, you must recall it using RCL M.

Note

This technique of solving for more than one unknown is most useful when certain variables will *not* be unknowns. The Solver equation of Example 12 is intended to be used only when *FATH* is known and *FT* and *M* are to be found. Solving for *M* will result in the message `SOLUTION NOT FOUND` since the direct Solver will attempt to isolate *M* and a division by zero error occurs. In general, this technique should *not* be employed when *all* variables in an equation will be calculated.

The next example combines several of the techniques you have learned so far to solve a practical problem.

Example 13: Complex Multiplication. To multiply two complex numbers $x = a + ib$ and $y = c + id$, use the formula

$$xy = (ac - bd) + i(bc + ad).$$

A Solver equation can be written that calculates the product, stores the real part of the product as a and the imaginary part as b, and leaves c and d unchanged. This makes the equation useful for chain calculations.

Equation:	Comments:
`0×L(R:A×G(C)−B×G(D))`	Stores the real part of the product xy in the intermediate variable R. The intermediate variable R is employed since we do not want to store the real part of the product in A yet. Before A can be assigned a new value, the current value of A is needed to calculate the imaginary part of the product. Notice that GET is used with C and D so that the menu of variables will be in the order A, B, C, D, and XY. This can be verified by looking at the equation as a whole and noting the order in which the values appear formally when scanning from left to right.
`+0×L(B:B×C+A×D)`	Stores the imaginary part of the product xy in B.

`+L(A:G(R))=XY`	Stores the real part of the product xy in A since the original value of A is no longer needed. GET is used with R since it is an intermediate variable and is not to appear in the menu. Notice that all the LET functions except this one are multiplied by zero. Thus, when XY is solved for, the Solver uses a direct solution method (XY appears formally only once) and effectively reduces the equation to $XY = R$. When XY is solved for, the real part of the product is displayed and the real and imaginary parts of the product are stored in A and B respectively. Notice that C and D are left unchanged.

Since the real part of the product is returned as the value of XY, this eliminates having to press RCL A after every calculation to see the real part of the product. To see the imaginary part, press RCL B.

An application with a several complex number functions has been developed in Chapter 2 of this book using the ideas in this equation.

Evaluation Order

As your Solver equations become increasingly more sophisticated, you may find that using LET and GET takes a bit of forethought to insure that the Solver assigns and recalls values in the order you intended. When calculating an unknown, your calculator effectively rearranges the equation and either isolates the variable in question and solves for it directly, or uses an iterative process. During rearrangement, the simple left-to-right order of evaluation may be disturbed.

For example, when `G(X)+Y=0xL(X:4+X)+4` is solved for Y, it is not obvious if the GET or LET is performed first. Actually, the Solver performs the LET *before* the GET in this equation.

Most ambiguities in using LET and GET can be avoided by observing the following guidelines:

1. Place all LET and GET functions on the same side of an equal sign.
2. Try to group the variable(s) you are calculating on the other side of the equals sign.

When these guidelines are followed, you can assume that the simple left-to-right evaluation process occurs.

Forcing Iteration

Although the idea of forcing iteration was introduced in your owner's manual, it bears repeating here. An equation such as $1=\sin x$ can be solved directly for x, but there are an infinite number of solutions to this equation given by $x = \pm(2n+1)\pi/2$ for $n = 0,1,2,....$ The Solver will find the root corresponding to the *principle value* of the sine function.

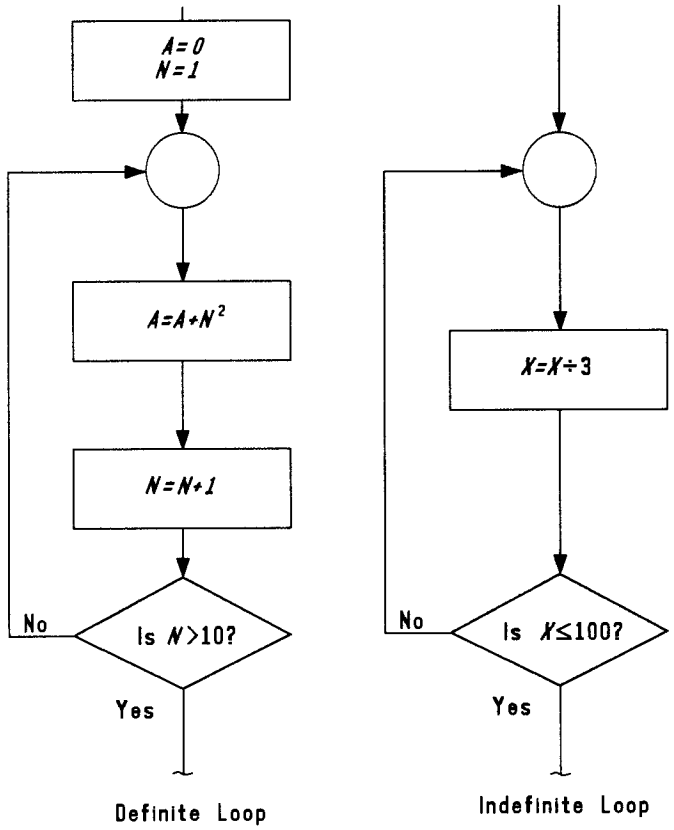
In general, the trigonometric functions operate using the principle value. If you are interested in a root that is *not* a principle value, you can re-write the equation in a mathematically equivalent form that forces an iterative solution. This allows you to enter initial guesses instructing the Solver to look for a root between the two bounds.

The equation above can be entered as the Solver equation `1=0xX+SIN(X)`. This forces an iterative search since X appears formally twice.

More on Using the Σ Function

Definite and Indefinite Loops

A loop is a technique used by computer programmers to repeat a certain section of instructions a number of times before continuing to other instructions. Often, the loop is executed a fixed number of times (a *definite* loop), while other times the loop repeats indefinitely until a certain condition is met. The second type of loop is referred to as an *indefinite* loop.



The definite loop illustrated in the figure will generate the sum of the squares of the integers 1 through 10. Notice that A is 0 initially, the loop counter N is 1 initially, and the loop counter is incremented by 1 with each pass through the loop. The definite loop is performed a *fixed* number of times (10 in this case). On the other hand, the indefinite loop repeats *indefinitely* until the desired condition ($X \leq 100$) is met.

By using the Σ function you can effectively include definite loops in your equations. In fact, the Σ function was *designed* to operate as a definite loop. Indefinite loops may also arise. While you are not able to construct a true indefinite loop for reasons that are explained below, you can effectively simulate one.

Simulating an Indefinite Loop

The Σ function is defined as follows:

Σ(*cv:c1:c2:s:alg*)

where the algebraic expression (*alg*) is evaluated and summed for values of the counter variable (*cv*). The counter variable starts with value *c1* and is incremented in steps of *s* to a final value of *c2*.

When the Σ function is first encountered in an equation, the Solver stores the step size *s* and the counter variable's initial and final values *c1* and *c2* in a special location in memory not accessible to the user. Any attempt to alter the values of *s*, *cv*, *c1*, or *c2* using the LET function causes the Solver to create *separate* variables of the *same* name. Since the value of these variables cannot be changed, a loop cannot be prematurely exited. This is precisely what makes construction of a true indefinite loop impossible. However, an indefinite loop can be simulated as shown in the next example.

Example 14: An Indefinite Loop. To simulate the indefinite loop shown in the previous figure, the Σ function can be performed until the desired condition ($X \leq 100$) is met. Then the loop can simply add zeros to this result on subsequent passes until the final value of the counter variable is reached. To avoid having too few or far too many loop repetitions, a way is needed to determine *in advance* the maximum number of loops necessary to meet the desired condition and to assign this value to *c2*.

For the example at hand, we must find the number of times n that X must be divided by 3. This is given by the equation

$$\frac{X_{initial}}{3^n} \leq 100.$$

If this is solved for n, we obtain the maximum number of loops needed to obtain the desired result ($X \leq 100$). Rearranging and taking the logarithms of both sides we have

$$\frac{X_{initial}}{100} = 3^n \rightarrow \ln\left(\frac{X_{initial}}{100}\right) = n \ln 3.$$

Solving for n, we obtain the final result

$$n = \frac{\ln\left(\frac{X_{initial}}{100}\right)}{\ln 3}.$$

This value of n is the value for *c2* that guarantees sufficient passes through the loop. The Solver equation is shown below.

Equation:	**Comments:**
`A=`	The variable we will calculate.
`Σ(N:1:`	The counter variable is N and is set initially to 1.
`IF(X<=300:1:` `LN(X÷100)÷LN(3)+1):`	This is the final value of the counter variable. The conditional check made by the `IF` insures that at least 1 loop will be performed (if X is less than 300, n is less than 1). In the event that $X > 300$, the result for n derived above is used with a small change: here we have added a 1 to the result. If this had not been done, the loop would only be performed $n-1$ times instead of n times.
`1:`	This is the step value; i.e., N is incremented by 1 each time the loop is repeated.
`IF(X<=100 AND N<>1:` `0:0×L(X:X÷3)))`	The body of the loop. If X is less than or equal to 100 and it is not the first pass through the loop ($N \neq 1$), a zero is added to the loop and X remains unchanged. If X is greater than 100 or $N = 1$ (first pass), the current value of X is divided by 3 and this result is assigned as the new value of X. Notice that the LET function is multiplied by zero causing the Σ function to have a value of zero. The 3 closing parentheses are needed to complete the LET function, the IF function, and the Σ function respectively.
`+X`	This term simply adds the final value of X to the value of the Σ function (which is zero as noted above) leaving the effective result $A = X$. This final value of X is returned as the solution for A.

A practical use of this Solver technique can be found in the application "Greatest Common Divisor and Least Common Multiple" in Chapter 2 of this book.

Using Trigonometric Functions

Equations involving trigonometric functions often demand that the variables be in radians rather than degrees. For example, in a branch of mathematics known as Fourier Transforms, the *sinc function* arises and is defined as

$$\text{sinc}x = \frac{\sin x}{x}.$$

Here, x must be in radians; however, it is often desirable to enter x in *either* radians or degrees. A convenient way to accomplish this is with a conditional check, illustrated in the following Solver equation:

```
SINC=IF(SIN(30)=.5:SIN(X)÷RAD(X):SIN(X)÷X).
```

Notice that the conditional check is true only when the calculator is in degrees mode. Although you must be aware of what mode the calculator is in when entering numbers in this Solver equation, this technique eliminates the need to always set radians mode and the need for two separate Solver equations (one for degrees and one for radians).

Note

The sinc function has the indeterminate value 0/0 at $x = 0$. By a technique of calculus known as *L'Hopital's Rule*, the sinc function can be shown to approach 1 as x approaches 0. Thus, the sinc function is defined as 1 at $x = 0$. To give a correct result for $x = 0$, the above Solver equation can be modified slightly to:

```
SINC=IF(X=0:1:IF(SIN(30)=.5:SIN(X)÷RAD(X):
SIN(X)÷X)).
```


In Conclusion

Although it is unlikely that you will want to use *every* application in this book, they represent operations that arise frequently in science and engineering. For this reason, you will probably want to keep several applications in your calculator's memory. To give yourself plenty of room to store and "CALC" the application Solver equations, we recommend that you delete the example equations created in this chapter after you have worked through them.

Recall from your owner's manual that Solver variables are "remembered" by the calculator for use in moving from one Solver equation to another. These variables consume a significant amount of calculator memory and should be periodically reviewed and deleted as described in your owner's manual.

2

Technical Application Equations

The topics included in this chapter contain Solver equations used in many science and engineering-related applications. In most cases, the examples have been chosen to reflect typical uses for such equations in engineering practice.

Greatest Common Divisor and Least Common Multiple

In some situations it is preferable to use fractions rather than decimals to express numbers (e.g. 1/17 rather than 0.05882). The techniques of adding, subtracting, multiplying, and dividing fractions require the ability to find *greatest common divisors* and *least common multiples*. Although simple in theory, this is often a tedious process.

The basic algorithm used in finding the greatest common divisor for two integers a and b is as follows:

1. If $b = 0$, GCD$(a,b) \leftarrow a$ and execution stops.
2. If $b \neq 0$, $z \leftarrow (a \bmod b)$, $a \leftarrow b$, and $b \leftarrow z$. Return to 1.

The least common multiple of a and b is found by

$$\text{LCM}(a,b) = \frac{ab}{\text{GCD}(a,b)}.$$

The equation uses an indefinite loop similar to the one that was discussed in "More on Using the Σ Function" in Part 1 of this book. Of particular importance is a formula that gives the maximum number of divisions M needed to arrive at a zero remainder. This formula is given as:

$$\text{For } n > a > b \geq 0, \quad M \approx 2.078 \ln(n) - .328$$

The development of this result as well as the algorithm above is in the reference cited following the examples.

The GCD and LCM Equation

Equation:	Comments:
`LCM/GCD:`	Equation name.
`0×L(A1:A)×L(B1:B)`	Stores initial values of a and b for use by LCM routine.
`+Σ(N:0:2.078× LN(A+1)-.328+1:1:`	Sets loop parameters, including the maximum number of divisions (loop repetitions) required.
`IF(B=0:0:0×L(Z: MOD(A:B))+0×L(A:B) +0×L(B:G(Z)))`	GCD algorithm.
`+A`	Value of GCD(a,b) after loop is completed.
`=IF(S(GCD):GCD: G(A1)×G(B1)÷LCM)`	Generates the proper result (*GCD* or *LCM*) depending upon what you are solving for.

Remarks on Using the Equation.

- The larger of the two numbers must be entered as A.
- The equation is designed to operate on positive integers only.
- You may wish to set your display format to "ALL" to eliminate extra trailing zeros since only integers are used.
- The equation assigns new values to A and B when calculating *LCM* or *GCD*; thus, the original values of A and B are lost.

Example Problems

Check to see that you have entered the equation described above properly, then press CALC to display the menu of variables.

Example 1. Find the greatest common divisor and least common multiple of 406 and 266.

Keys:	Display:	Description:
■ MODES ALL *		Set the display format to ALL to eliminate trailing zeros.
406 A	A=406	Stores a.
266 B	B=266	Stores b.
GCD	GCD=14	Greatest common divisor.
406 A	A=406	Re-enters a.
266 B	B=266	Re-enters b.
LCM	LCM=7,714	Least common multiple.

Example 2: Addition of Fractions. Express $\frac{71}{494}+\frac{39}{1026}$ as a single fraction. Make sure that your display is still set to "ALL" as shown in the previous example.

Keys:	Display:	Description:
1026 A	A=1026	Stores a.
494 B	B=494	Stores b.
LCM	LCM=13,338	Least common multiple (denominator of final fraction).
÷ 494 × 71 + (RCL LCM ÷ 1026 × 39 B	B=2,424	Assigns the numerator of final fraction i.e., $\frac{13338}{494}\times 71+\left(\frac{13338}{1026}\times 39\right)$, to b.
RCL LCM STO A	A=13,338	Re-enters a.
GCD	GCD=6	Greatest common divisor.

Thus,

$$\frac{71}{494}+\frac{39}{1026}=\frac{2424/6}{13338/6}=\frac{404}{2223}.$$

* To set the display format to ALL on the HP-19B, press DISP ALL.

Reference

Knuth, Donald E., *The Art of Computer Programming*, Vol. 2, Addison-Wesley, Reading, MA, 1969.

Numerical Integration

Simpson's Rule is widely used to approximate definite integrals. This is due to its simplicity, good results, and ease of implementation. Simpson's Rule essentially divides the area to be integrated into an even number of subintervals and interpolates a quadratic polynomial to $f(x)$ at the top of each subinterval.

For integrals of the form

$$\int_a^b f(x)\,dx$$

the approximation for $2n$ subintervals is given by:

$$S_{2n} = \frac{(b-a)/2n}{3}[f_0+4f_1+2f_2+4f_3+2f_4+\cdots+2f_{2n-2}+4f_{2n-1}+f_{2n}].$$

This approximation is valid if the integral meets the following conditions:

1. The limits of integration are finite.
2. For all $a \le x \le b$, $f(x)$ is both finite and defined.

A judicious change of variables can sometimes transform an integral violating these conditions into an acceptable form. For example, consider the improper integral

$$\int_1^\infty dx/(1+x^{64}).$$

The change of variable $u = 1/x$ transforms this integral to

$$\int_0^1 u^{62}du/(1+u^{64})$$

which is easily evaluated.

Errors in approximating definite integrals with Simpson's Rule arise in two ways:

1. Error due to the quadratic polynomial substituted for $f(x)$ in each subinterval.
2. Round-off error during calculations due to the limited precision of your calculator.

While the second source of errors can never be completely eliminated, the first source of errors can generally be made arbitrarily small by choosing a large enough number of subintervals. While increasing the number of subintervals will usually improve the accuracy of the results, it also increases the computation time needed by the calculator.

The Integration Equation

Equation:	Comments:
`INTEGRATE:`	Equation name.
`I=L(H:(A-B)÷(-2×N))`	Interval size set to $(b-a)/2n$.
`÷3×Σ(R:0:2×N:1:0×L(X:A+R×G(H))`	Evaluates $f(x)$ at $2n+1$ evenly spaced points.
`+(SGN(R)+SGN(2×N-R)+2×MOD(R:2))×(FX))`	Multiplies $f(x)$ by proper constant.

Remarks on Using the Equation.

- When integrating trigonometric equations, radians mode *must* be set.
- The function being integrated should be written with x as the independent variable.
- The function being integrated is used in place of *FX* in the Solver equation above.

Example Problems

In preparation for the first example, enter the equation described above using the following function in place of *FX*.

$$f(x)=36\pi\left[\left(1-\frac{x^2}{81}\right)\left(1+\frac{16x^2}{324-4x^2}\right)\right]^{\frac{1}{2}}$$

The Solver equation should now look like this:

```
INTEGRATE:I=L(H:(A-B)÷(-2×N))÷3
×Σ(R:0:2×N:1:0×L(X:A+R×G(H))+(SGN(R)
+SGN(2×N-R)+2×MOD(R:2))×(36×PI×SQRT(1
-SQ(X)÷81)×SQRT(1+(16×SQ(X))÷(324-
4×SQ(X))))
```

When your equation matches this one, press [CALC] to display the menu of variables.

Example 1: Surface Area Computation. A new water tank is to be constructed by Erisman Industries. The tank has a circular cross-section when viewed from above. When viewed from the side its cross-section is that of an ellipse with height 18 ft. and width 36 ft. How many square feet of steel should Mr. Erisman plan on using for the tank?

The shape of the tank can be generated by revolving the ellipse

$$\frac{x^2}{18^2}+\frac{y^2}{9^2}=1$$

about the y-axis. The surface area integral is then

$$A=2\pi\int_a^b f(y)\sqrt{1+[f'(y)]^2}\,dy.$$

Substituting the values for the problem at hand we obtain

$$36\pi\int_{-9}^{9}\left[\left(1-\frac{y^2}{81}\right)\left(1+\frac{16y^2}{324-4y^2}\right)\right]^{\frac{1}{2}}dy.$$

Since the integrand becomes infinite at both endpoints ($y=\pm 9$), we will adjust our limits to $y=\pm 8.999$ to avoid an error condition when the Solver evaluates the equation. We will use 40 subintervals to approximate the integral.

Keys:	Display:	Description:
8.999 [+/-] [A]	A=-8.9990	Stores *a*.
8.999 [B]	B=8.9990	Stores *b*.
20 [N]	N=20.0000	Stores *n* (recall that the approximation is for 2*n* subintervals or 40 in this case).
[I]	I=2,809.2376	Approximate surface area in square feet.

For comparison, the exact result is

$$2\pi(9)(18)\left[\frac{18}{9}+\frac{9}{\sqrt{18^2-9^2}}\ln\left(\frac{18+\sqrt{18^2-9^2}}{9}\right)\right]=2809.68999 \text{ square feet.}$$

The numerical estimate is accurate to within .016%.

Example 2: Broadcast Signal Coverage. An AM radio station radiates a signal from its directional antenna array in the shape of the cardioid $r=30(1+\cos\theta)$, where θ is the radial angle around the antenna array. If r is measured in miles, how many square miles of coverage does this radio station have?

The equation for area in polar coordinates is:

$$A = \frac{1}{2}\int_{\theta_1}^{\theta_2} r^2\, d\theta.$$

For this problem we have

$$A = \int_{0}^{2\pi} 450(1+\cos\theta)^2\, d\theta.$$

Enter this equation in the INTEGRATE equation by replacing θ with x. It is not necessary to re-type the entire equation. Simply edit the existing INTEGRATE equation. Your equation should look like this:

```
INTEGRATE:I=L(H:(A-B)÷(-2×N))÷3
×Σ(R:0:2×N:1:0×L(X:A+R×G(H))+(SGN(R)
+SGN(2×N-R)+2×MOD(R:2))×(450×SQ(1+COS(X))))
```

When your equation matches this one, press [CALC] to display the menu of variables. Evaluate the area using 40 subintervals.

Keys:	Display:	Description:
■ [MODES] [MORE] [D/R] [EXIT] *		Set radians mode.
0 [A]	A=0.0000	Stores a.
2 [×] ■ [π] * [=] [B]	B=6.2832	Stores b.
20 [N]	N=20.0000	Stores n.
[I]	I=4,241.1501	Approximate area in square miles.

* To set radians mode on the HP-19B, press: ■ [MODES] [D/R] [EXIT].

The exact area is $6\pi\ (30/2)^2 = 4241.15008234$ square miles.

Reference

Kaplan, W., *Advanced Mathematics for Engineers*, Addison-Wesley, Inc., Reading, MA, 1981.

* To enter π from the HP-19B, press: ■ [MATH] [PI] [EXIT].

Numerical Differentiation

The numerical differentiation Solver equation developed below calculates the numerical value of a function's derivative at a point. Using the features of your Solver, minimums and maximums of functions can also be found.

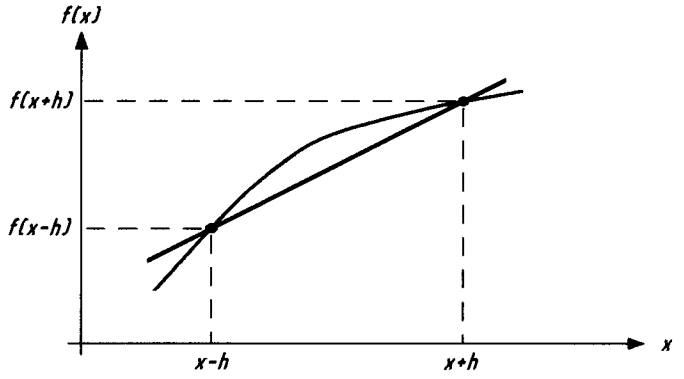
The defining equation for the derivative is the limit of the *difference quotient*:

$$f'(x)=\lim_{h\to 0}\frac{f(x+h)-f(x)}{h}.$$

Another way of expressing a derivative is with the equation

$$f'(x)=\lim_{h\to 0}\frac{f(x+h)-f(x-h)}{2h}.$$

This equation can be shown to approach its limit faster than the first equation and for this reason will be used to approximate $f'(x)$ for suitably small h. (More about the choice of h later.) The figure below shows that this approximation is the slope of the secant line through $f(x+h)$ and $f(x-h)$.



We can use the previous results and an increment of $\frac{h}{2}$ to approximate $f''(x)$ as

$$f''(x)\approx\frac{f'(x+h/2)-f'(x-h/2)}{(2)(h/2)}.$$

This can be simplified to

$$f''(x)\approx\frac{f(x+h)-2f(x)+f(x-h)}{h^2}$$

which will be used since it involves only the function (not its first derivative).

Choosing h

The HP-27S and HP-19B store numbers using 12-digit precision. This means that $2+10^{-12}$ and $2-10^{-12}$ will both be rounded and stored as 2. Thus, if you are calculating

$$f'(2)\approx\frac{f(2+h)-f(2-h)}{h}$$

you will *always* obtain a zero result for $h=1\times10^{-12}$. In general, x and h cannot be separated by more than 12 orders of magnitude.

In addition to the effects of round-off error on the *argument* of f, another equally important consideration in choosing h is the effect of round-off error on the *result*. If h is too small, $f(x+h)$ and $f(x-h)$ may only differ in decimal places beyond the 12-digit precision of your calculator. This rounding of $f(x+h)$ and $f(x-h)$ may cause $f(x+h)-f(x-h)$ to deviate significantly from its actual value, thus affecting the approximation of $f'(x)$. For example, if $f(x)=\sqrt{x}$ using $h=1\times10^{-11}$ we have

$$f'(1)\approx\frac{\sqrt{1+10^{-11}}-\sqrt{1-10^{-11}}}{2\times10^{-11}}=0.25.$$

The true value is 0.5. The error here is because $\sqrt{1+10^{-11}}$ is rounded to 1. Values for h in the range $\frac{x}{10^5}\le h\le\frac{x}{10^4}$ give good results for $f'(x)$; similarly, for $f''(x)$, $\frac{x}{10^4}\le h\le\frac{x}{10^3}$ gives good results.

The Differentiation Equation

The listings below outline Solver equations for differentiation.

Equation:	Comments:
`DY/DX:`	Equation name.
`F'X=0xL(Q:X)`	Create local variable so x will not be lost.
`+Σ(N:-1:1:2:0xL (X:G(Q)+NxH)+ Nx(FX)÷(2xH))`	Equivalent to $\frac{f(x+h)-f(x-h)}{2h}$ where *FX* is replaced by the function to be differentiated.
`+0xL(X:G(Q))`	Restores value of x.

Equation:	Comments:
`D2Y/DX2:`	Equation name.
`F''X=0xL(Q:X)`	Create local variable so x will not be lost.
`+Σ(N:1:3:1:0xL(X: G(Q)+Hx(2-N))+ (-1)^(N+1)x(2- MOD(N:2))x(FX)÷SQ(H))`	Equivalent to $\frac{f(x+h)-2f(x)+f(x-h)}{h^2}$.
`+0xL(X:G(Q))`	Restores value of x.

Remarks on Using the Equations.

- In each equation, *FX* is replaced with the function to be differentiated.
- The functions to be differentiated must be entered with x as the independent variable.
- Trigonometric functions *must* have radians mode set for proper results.

Example Problems

Key in the Solver equations listed with

$$f(x)=y_0-\frac{1}{\rho}ln(\cosh x\sqrt{\rho g})$$

in preparation for the first example. Your equations should look like this:

```
DY/DX:F'X=0xL(Q:X)+Σ(N:-1:1:2:0xL(X:G(Q)
+NxH)+Nx(Y0-(1÷P)xLN(COSH(
XxSQRT(PxG)))÷(2xH))+0xL(X:G(Q))
```

```
D2Y/DX2:F''X=0xL(Q:X)+Σ(N:1:3:1:0xL(X:G(Q)+H
x(2-N))+(-1)^(N+1)x(2-MOD(N:2))x(Y0-(1÷P)
xLN(COSH(XxSQRT(PxG)))÷SQ(H))+0xL(X:G(Q))
```

When your equations match these, press CALC to display the menu of variables.

Example 1: Freefall of Heavy Bodies. As part of a TV advertising campaign for a new pick-up truck, creative genius Will Selmore proposes to have several pick-ups dropped by parachute into a large field for the video portion of the ad. To ensure that the trucks will not be damaged on impact they must be traveling no faster than 200 ft/sec when their parachutes open. If the trucks are dropped from 6,000 ft. and allowed to freefall for 10 seconds, will they land safely? Also, find the acceleration at the instant when the parachutes open.

The equation governing freefall of a heavy body under the influence of gravity and air resistance is:

$$y(t)=y_0-\frac{1}{\rho}ln(\cosh t\sqrt{\rho g})$$

where

g = 32 ft/sec (acceleration due to gravity)
ρ = .001 (drag coefficient)
y_0 = 6000 ft. (initial height)
$y(t)$ = position at time t

Since you have already loaded this function into both the 1st and 2nd derivative Solver equations, the velocity at $t = 10$ seconds can be found simply by evaluating dy/dx at $x = 10$. Use $h = 0.001$.

Keys:	Display:	Description:
10 [X]	X=10.0000	Stores x.
.001 [H]	H=0.0010	Stores h.
6000 [Y0]	Y0=6,000.0000	Stores y_0.
.001 [P]	P=0.0010	Stores ρ.
32 [G]	G=32.0000	Stores g.
[F'X]	F'X=−169.1611	Velocity in ft/sec at $t = 10$ seconds.

The minus sign indicates the trucks are falling downward. The actual velocity is

$$dy/dt = v = -\sqrt{g/\rho}\tanh t\sqrt{\rho g} = -169.161141610$$

The numerical estimate agrees with the actual velocity to 4 decimal places. Since the speed is less than 200 ft/sec the trucks will land safely.

The acceleration is found by using the Solver equation D2Y/DX2. Move the pointer in your Solver to this equation and press [CALC]. The values from the previous equation are saved. Simply change h to 0.01 and solve for $f''(x)$.

Keys:	Display:	Description:
.01 [H]	H=0.0100	Stores new value of h.
[F"X]	F"X=−3.3845	Acceleration in ft/sec^2.

The true acceleration is

$$\frac{d^2y}{dt^2} = -g + g\tanh^2 t\sqrt{\rho g} = -3.38450816930 \text{ ft/sec}^2.$$

The estimate agrees to 4 decimal places.

Minimum/Maximum Problems

An extremely powerful application of the derivative is that of finding extrema of functions. Local maximums and minimums potentially occur where $f'(x)$ is 0 or undefined. If an extrema exists, the sign of the second derivative indicates the type of extrema (negative for local maximums and positive for local minimums).

Example 2: Minimum Field Strength. For the design of a vertical broadcasting tower, radio engineer Ann Tenor wants to find the angle from the tower at which the relative field intensity is most negative. The relative field intensity created by the tower is given by:

$$E = \frac{\cos(2\pi l\cos\theta) - \cos(2\pi l)}{\left[1-\cos(2\pi l)\right]\sin\theta}$$

where E is the relative field intensity, l is the antenna height in wavelengths, and θ is the angle from vertical in radians. The height is 0.6 wavelengths for her design.

The desired angle is one at which the derivative of the intensity with respect to θ is zero. Edit both existing derivative equations (DY/DX and D2Y/DX2) replacing the old $f(x)$ with the function for E above (remember to enter the function in terms of x instead of θ). Your equations should look like this:

```
 DY/DX:F'X=0×L(Q:X)+Σ(N:-1:1:2:0×L(X:G(Q)
+N×H)+N×((COS(2×PI×L×COS(X))
-COS(2×PI×L))÷((1-COS(2×PI×L))×SIN(X)))
÷(2×H))+0×L(X:G(Q))
```

```
D2Y/DX2:F''X=0xL(Q:X)+Σ(N:1:3:1:0xL(X:G(Q)
+Hx(2-N))+(-1)^(N+1)x(2-MOD(N:2))x((COS
(2xPIxLxCOS(X))-COS(2xPIxL))÷((1-COS(2xPIxL))
xSIN(X)))÷SQ(H))+0xL(X:G(Q))
```

When your equations match these, set radians mode, adjust your Solver pointer to DY/DX, and press [CALC] to display the menu of variables. Solve for $\frac{dE}{d\theta}=0$ using an h of .001.

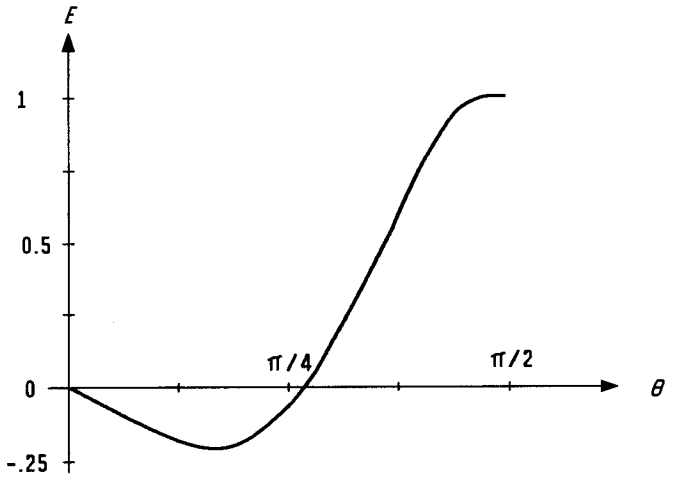
Keys:	Display:	Description:
■ [MODES] [MORE] [D/R] [EXIT] *		Set radians mode, if necessary.
■ [CLEAR DATA]	0.0000	Sets all variables equal to zero, including $F'(X)$.
.6 [L]	L=0.6000	Stores l.
.001 [H]	H=0.0010	Stores h.
[X]	X=0.4899	Angle in radians where extrema occurs.

Now see if this corresponds to a local maximum or minimum by executing D2Y/DX2. The values from the previous calculation are retained, but you should use a slightly larger h of .01. Position your pointer to this equation and press [CALC].

Keys:	Display:	Description:
.01 [H]	H=0.0100	New value of h.
[F"X]	F"X=2.7105	Value of $\frac{d^2E}{d\theta^2}$.

* To set radians mode on the HP-19B, press: ■ [MODES] [D/R] [EXIT].

The positive value indicates that a local minimum was found. To ensure that this is an *absolute* minimum on the interval $0° \le \theta \le 90°$ you should check for other zeros of the first derivative. You will find an absolute maximum at $\theta=90°$ and no other extrema in the interval, as shown in the figure below. Thus, the most negative relative field intensity occurs at θ = .4899 radians = 28.0692°. For comparison the exact minimum occurs at $\theta=28.068°$.



Example 3: Minimizing Cost per Part. A production line supervisor for the Rainy Day Sprinkler Company must determine the optimal number of machines to install for making 500,000 plastic housings for a new line of lawn sprinklers. It costs \$6,750 to set up each machine initially and \$250+ \$37.60n to run n machines for 1 hour. Each machine is capable of producing 73 parts/hour. The parts must be ready in 6 weeks and the machines can run at most 6 days a week for 16 hours a day.

The optimal number of machines occurs where the cost per part is minimum. The set-up cost is $6750n/500000$ (\$/part). The remaining cost is the hourly cost per part, and is given by

$$\frac{1}{73n}\ (\text{hours/part}) \times (250+37.60n)\ (\$/\text{hour}) = \frac{250+37.60n}{73n}\ (\$/\text{part}).$$

The total cost is then

$$\frac{6750n}{500000} + \frac{250+37.60n}{73n} (\$/\text{part}).$$

Edit the DY/DX equation replacing the old function of x with the one above (using x in place of n). It should look like this:

```
DY/DX:F'X=0×L(Q:X)+Σ(N:-1:1:2:0×L(X:G(Q)
+N×H)+N×(250÷(73×X)+37.6÷73
+6750×X÷500000)÷(2×H))+0×L(X:G(Q))
```

Press [CALC] to display the menu of variables, and solve for $f'(x)=0$ using an h of .001. Instruct the Solver to look for a zero in the *positive* direction (you cannot have a negative number of machines) by entering initial guesses of 1 and 50. The Solver will search between these two bounds for a zero.

Keys:	**Display:**	**Description:**
.001 [H]	H=0.0010	Stores h.
0 [F'X]	F'X=0.0000	Stores $f'(x)$.
1 [X] 50 [X] [X]	X=15.9273	Stores search boundaries and locates minimum.

Since the number of machines must be an integer, we round this to 16. You may wish to verify that this is indeed an absolute minimum over the interval 1 to 50 and that the function increases without bound for $x > 16$. For $n = 15$ and $n = 17$ the cost per part is 94.6 cents; for $n = 16$ the cost per part is 94.5 cents. This small difference amounts to a $500 savings for 500,000 parts. Finally, to insure the validity of this answer, verify that 16 machines will be able to produce 500,000 parts in the allotted time.

6 (wks) × 6 (dys/wk) × 16 (hrs/dy) × 73 (parts/machine-hr) × 16 (machines) = 672,768 parts.

Thus, 16 machines can meet the time constraint with the least cost per part.

References

Edwards, Penney, *Calculus and Analytic Geometry*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1982.

Kaplan, W., *Advanced Mathematics for Engineers*, Addison-Wesley, Inc., Reading, MA, 1981.

Factors and Primes

The factors and primes Solver equation will find all the prime factors of a positive integer n.

The algorithm for this equation selects a trial divisor d and tests d as a factor of n. If d divides n, then $n \leftarrow n/d$ and d is tested as a factor of the new n. If d does not divide n, a new d is selected. The process continues until $d > \sqrt{n}$, at which point n is returned as the final factor. The trial divisor d takes on the values 2, 3, 5, and 7; then for $d > 10$, d takes on those values that satisfy $(d-10) \bmod 30 = 1, 3, 7, 9, 13, 19, 21$, or 27. Thus, in the first cycle of 30 integers from 11 to 40, d assumes the values 11, 13, 17, 19, 23, 29, 31, 37. This technique eliminates from the test those values of d $(d > 10)$ that are divisible by 2, 3, or 5.

To translate this algorithm into a suitable Solver equation we use the following techniques:

1. Use nested IF functions to test for a factor of 2, 3, 5, or 7.
2. Use a final nested IF as a loop that looks for prime factors ≥ 11.

The Factors and Primes Equation

The following Solver equation uses the algorithm described above to determine the prime factors of a number.

Equation:	Comments:
`FACTOR:`	Equation name.
`FACT=0×L(E:N)`	Stores N in intermediate variable E so that initial value of N will not be lost.
`+IF(MOD(N:2)=0:` `0×L(E:2):`	If 2 is a factor store 2 in E, otherwise continue.
`IF(MOD(N:3)=0:` `0×L(E:3):`	If 3 is a factor store 3 in E, otherwise continue.
`IF(MOD(N:5)=0:` `0×L(E:5):`	If 5 is a factor store 5 in E, otherwise continue.
`IF(MOD(N:7)=0:` `0×L(E:7):`	If 7 is a factor store 7 in E, otherwise continue.
`L(J:0)`	Initializes "first factor" flag J to zero.

`+Σ(D:11:SQRT(N):2:`
`IF(G(J)=1:0:`
`IF(L(C:MOD(D-10:30))`
`=1 OR G(C)=3 OR`
`G(C)=7 OR G(C)=9 OR`
`G(C)=13 OR G(C)=19`
`OR G(C)=21 OR`
`G(C)=27:`
	If the first prime factor has already been found ($J=1$) zeros are added. If not, the prime divisors greater than 10 are generated.

`IF(MOD(G(E):D)=0:`	Tests to see if current value of D is a prime factor.
`0×L(J:1)×L(E:D):`	If so, value of D is saved and "first factor" flag J is set.
`0):`	If not, loop gets next value of D.
`0)))))))`	Other argument of the first IF function. Only occurs if no prime factors are found.
`+G(E)+L(J:0)×` `L(N:N÷G(E))`	Returns the value of the first prime factor encountered, clears the "first factor" flag J, and adjusts N to its new value (N_{old}/D).

Remarks on Using the Equation.

- n must be an integer ≥ 1.
- To eliminate extra trailing zeros, set your display format to "ALL".
- You should expect long execution times for very large integers.

Example Problems

Enter the Solver equation FACTOR, taking special care to include the correct number of parentheses and to put spaces around the AND functions. When your equation matches the equation listed, press ≡CALC≡, then set your display to ALL. You should see only ≡FACT≡ and ≡N≡ on the menu of variables.

Example 1. Find all the prime factors of 924.

Keys:	Display:	Description:
■ MODES ≡ALL≡ *		Set the display to ALL.
924 ≡N≡	N=924	Stores number to be factored.
≡FACT≡	FACT=2	First factor.
≡FACT≡	FACT=2	Second factor.
≡FACT≡	FACT=3	Third factor.
≡FACT≡	FACT=7	Fourth factor.
≡FACT≡	FACT=11	Fifth factor.
≡FACT≡	FACT=1	A result of 1 indicates complete factorization.

Thus, $924 = 2 \times 2 \times 3 \times 7 \times 11$.

Example 2. Find all the prime factors of 3623.

Make sure that your display is still set to ALL as in the previous example.

Keys:	Display:	Description:
3623 ≡N≡	N=3,623	Stores number to be factored.
≡FACT≡	FACT=3,623	Indicates 3623 has no prime factors...it is prime itself.

* To set the display to ALL on the HP-19B, use the following keystrokes: DISP ≡ALL≡.

Vector Operations

Because vectors arise frequently in engineering computations, it is convenient to have several vector operations available. The most common of these are the cross product, the dot product, the magnitude of a vector, and the angle between 2 vectors. Given the vectors $V_1 = x_1\boldsymbol{i} + y_1\boldsymbol{j} + z_1\boldsymbol{k}$ and $V_2 = x_2\boldsymbol{i} + y_2\boldsymbol{j} + z_2\boldsymbol{k}$ defined in a 3-dimensional rectangular coordinate system, the definitions are as follows:

Cross product:

$$V_1 \times V_2 = \begin{vmatrix} \boldsymbol{i} & \boldsymbol{j} & \boldsymbol{k} \\ x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \end{vmatrix}$$

Dot product:

$$V_1 \cdot V_2 = x_1x_2 + y_1y_2 + z_1z_2$$

Magnitude:

$$\left|V_1\right| = \sqrt{x_1^2 + y_1^2 + z_1^2}$$

Angle between vectors:

$$\gamma = \cos^{-1}\frac{V_1 \cdot V_2}{\left|V_1\right|\left|V_2\right|}$$

The Vector Operations Equation

Since it is convenient to have all the functions defined above available on a single menu, nested IF S functions are used. The equation is as follows:

Equation:	Comments:
`VECTOR:`	Equation name.

`IF(S(CROSS):CROSS:`
`IF(S(DOT):DOT:`
`IF(S(ANG):ANG:MAG)))=`
	Nested IF's to assign result to proper variable.

`0×L(M:SQRT(SQ(X1)+` `SQ(Y1)+SQ(Z1)))`	Stores magnitude of V_1 in intermediate variable *M*.

`+IF(S(DOT) OR`
`S(ANG):0×L(C:X1×X2+`
`Y1×Y2+Z1×Z2)`
	If *DOT* or *ANG* are desired, compute dot product and store in intermediate variable *C*.

`+IF(S(ANG):`
`ACOS(G(C)÷(G(M)×`
`SQRT(SQ(X2)+SQ(Y2)+`
`SQ(Z2))))):G(C))`
	If solving for angle, compute angle; if solving for dot product, recall *C*.

`:IF(S(MAG):G(M):`	If dot product or angle is not being solved for, check if magnitude is sought. If so, recall *M*.

`0×L(A:Z1×X2-X1×Z2)×`
`L(B:X1×Y2-Y1×X2)+`
`L(X1:Y1×Z2-Z1×Y2)+`
`0×L(Y1:G(A))+`
`0×L(Z1:G(B)))`
	If not, then cross product is the only possibility left and it is computed. The intermediate variables *A* and *B* will store the y- and z-components of the resulting vector until equation finishes using the original components of V_1.

Remarks on Using the Equation.

- *DOT*, *MAG*, and *ANG* leave the x, y, and z-components of their arguments unchanged. You can then do computations using the same vectors without re-entering the vector components.
- The magnitude of V_1 is calculated when the *MAG* function is executed.
- *CROSS* returns the resulting vector as V_1. The x-component is displayed and the other components can be viewed by using the [RCL] key. This feature is useful for chain calculations.
- For two dimensional vectors, simply consider that the k component does not exist, i.e. enter 0 for the z's.

Example Problems

Key in the equation described above, making sure to put spaces around the OR function. When you finish, press [CALC] to display the menu of variables.

Example 1: Force on a Transmission Line. A long straight transmission line carrying a DC current of 200 A in the direction $\boldsymbol{d} = 10\boldsymbol{i} + 7.3\boldsymbol{j} - 4.1\boldsymbol{k}$ (m) is immersed in a uniform magnetic field $\boldsymbol{B} = .008\boldsymbol{i} - .0015\boldsymbol{j} - .0049\boldsymbol{k}$ where $\boldsymbol{B}$ is in Wb/m^2. Find the force per unit length (N/m) acting on the wire.

The force per unit length is given by

$$\frac{\boldsymbol{F}}{l} = I\boldsymbol{d} \times \boldsymbol{B}$$

where l is the length of the wire and I is the DC current in the wire.

Keys:	Display:	Description:
10 [×] 200 [X1]	X1 = 2,000.0000	Stores x-component of 200 $\boldsymbol{d}$.
[MORE] 7.3 [×] 200 [Y1]	Y1 = 1,460.0000	Stores y-component of 200 $\boldsymbol{d}$.
4.1 [+/-] [×] 200 [Z1]	Z1 = −820.0000	Stores z-component of 200 $\boldsymbol{d}$.
.008 [X2]	X2 = 0.0080	Stores x-component of $\boldsymbol{B}$.
.0015 [+/-] [Y2]	Y2 = −0.0015	Stores y-component of $\boldsymbol{B}$.
.0049 [+/-] [Z2]	Z2 = −0.0049	Stores z-component of $\boldsymbol{B}$.
[MORE] [CROSS]	CROSS = −8.3840	x-component of force per unit length
[MORE] [RCL] [Y1]	Y1 = 3.2400	y-component of force per unit length
[RCL] [Z1]	Z1 = −14.6800	z-component of force per unit length

Thus, the force per unit length acting on the wire is $-8.384\boldsymbol{i} + 3.24\boldsymbol{j} - 14.68\boldsymbol{k}$ (N/m).

Example 2: Work Done by a Space Probe. A space probe must be repositioned by flight controllers on earth so that it can achieve a better viewing angle of one of Saturn's rings. To place the probe in a proper position for obtaining photographs, it must move to a position $100\boldsymbol{i} + 534\boldsymbol{j} + 378\boldsymbol{k}$ (km) from its present position. The probe is acted upon by the gravitational fields of several bodies which exert a total force of $\boldsymbol{F} = -5.2\boldsymbol{i} + 3.4\boldsymbol{j} - 1.6\boldsymbol{k}$ (N). Find the energy expended by the probe in moving to the desired position.

The work done is given by the formula

$$W = \boldsymbol{F} \cdot \boldsymbol{d}$$

where $\boldsymbol{d}$ is the vector of motion.

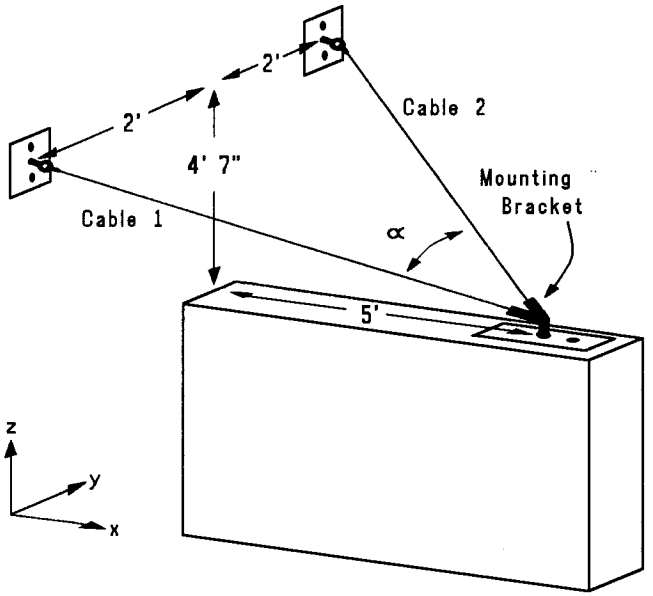
Set the display format to "FIX 2" for this example.

Keys:	Display:	Description:
■ MODES FIX 2 INPUT *		Set the display to FIX 2.
100 X1	X1 = 100.00	Stores x-component of distance vector.
MORE 534 Y1	Y1 = 534.00	Stores y-component of distance vector.
378 Z1	Z1 = 378.00	Stores z-component of distance vector.
5.2 +/- X2	X2 = −5.20	Stores x-component of force vector.
3.4 Y2	Y2 = 3.40	Stores y-component of force vector.
1.6 +/- Z2	Z2 = −1.60	Stores z-component of force vector.
MORE DOT	DOT = 690.80	Dot product.

Thus, the energy expended is 690.8 kJ.

* To set the display to FIX 2 on the HP-19B, use the following keystrokes: DISP FIX 2 INPUT.

Example 3: Angle for a Mounting Bracket. A mounting bracket for supporting a sign is to be constructed as shown in the figure below. What angle α should be used for the dimensions given? Also, find the length of cable needed.



Represent the cables as vectors going *from* the wall *to* the brackets and compute the angle α between them for the bracket angle. For the length of cable needed, find the magnitude of the vector representing one cable. By symmetry, the total cable length is twice this.

For this example set the display mode to "FIX 2" and the angular mode to degrees.

Keys:	Display:	Description:
■ MODES FIX 2 INPUT *		Set the display to FIX 2, if necessary.
■ MODES MORE D/R EXIT †		Set degrees mode, if necessary.
5 X1	X1 = 5.00	Stores x-coordinate of cable 1.
MORE STO X2	X2 = 5.00	Stores x-coordinate of cable 2.
2 Y2	Y1 = 2.00	Stores y-coordinate of cable 1.
+/- Y2	Y2 = −2.00	Stores y-coordinate of cable 2.
7 ÷ 12 + 4 = +/- Z1	Z1 = −4.58	Converts 4 feet 7 inches to decimal feet and stores as z-coordinate of cable 1.
STO Z2	Z2 = −4.58	Stores −4.58 feet as the z-coordinate of cable 2.
MORE ANG	ANG = 32.86	Angle between cables in degrees.
MAG	MAG = 7.07	Length of one cable in feet. (Double this for total cable needed.)

* To set the display to FIX 2 on the HP-19B, use the following keystrokes: DISP FIX 2 INPUT.

† To set degrees mode on the HP-19B, use the following keystrokes: ■ MODES D/R EXIT.

Thus, the total length of cable needed is about 14 feet 2 inches and the bracket should have an angle of about 33°.

References

Meriam, J.L., *Engineering Mechanics*, Vol. 1 and 2, John Wiley and Sons, New York, 1980.

Resnick, Robert and David Halliday, *Fundamentals of Physics*, 2nd Edition, John Wiley and Sons, Inc., New York, 1981.

Complex Number Operations

Given the complex numbers $x = a + ib = re^{i\theta}$ and $y = c + id = me^{i\phi}$, the following operations will be implemented:

Addition: $x + y = (a + b) + i(c + d)$

Subtraction: $x - y = (a - b) + i(c - d)$

Multiplication: $xy = (ac - bd) + i(ad + bc)$

Division: $\frac{x}{y} = \frac{(a + ib)(c - id)}{c^2 + d^2}$

Logarithm: $\ln(x) = \ln(re^{i\theta}) = \ln(r) + i\theta$

Power: $x^y = e^{\ln x^y} = e^{y \ln x}$

Inverse: $\frac{1}{x} = \frac{a - ib}{a^2 + b^2}$

Swap: exchanges x and y

The Complex Operations Equation

The Solver equation listed here implements the operations described above.

Equation:	Comments:
`COMPLEX:`	Equation name.
`IF(S(SWAP):`	Is a swap desired?
`0×(L(R:RX)+L(I:IX))`	If so, store the real and imaginary parts of x in the intermediate variables R and I.
`-L(RX:RY)+0×(L(IX:IY)`	Store the real and imaginary part of y in x.
`+L(RY:G(R))` `+L(IY:G(I)))`	Store the original contents of x in y.
`+SWAP:`	End of execution when *SWAP* is solved for. Mulitiplication by zero has been used to assign proper results to variables. This leaves the effective equation $-SWAP + RX = 0$. This is solved directly to obtain $SWAP = RX$.
`IF(S(MUL) OR S(DIV):`	Are either multiplication or division desired? (Note the spaces around OR.)
`IF(S(DIV):`	If so, is division desired?
`0×(L(RX:RX` `÷SQ(RADIUS(RY:IY)))`	If division is desired, execution comes here. Real part of x (RX) is divided by denominator in division definition above. This becomes new RX.
`+L(IX:IX` `÷SQ(RADIUS(RY:IY)))`	Analagous operation is done on imaginary part of x (IX).
`+L(R:RX×RY+IX×IY)`	R is used as an intermediate variable to store the final value of RX (old value of RX is still needed).
`+L(IX:IX×RY-RX×IY))`	IX given its final value. Note that a 2nd closing parenthesis encloses those functions that are multiplied by 0.
`-L(RX:G(R))`	Assigns RX its proper value.

`+DIV:`	End of execution when *DIV* is solved for. Multiplication by zero has been used to assign proper values to variables. This leaves the equation $-RX+DIV=0$. This is solved directly to obtain $DIV=RX$.
`0×(L(R:RX×RY-IX×IY)`	If multiplication is desired, execution jumps to here. This line uses *R* as an intermediate variable to store the final value of *RX* (old value of *RX* is still needed).
`+L(IX:IX×RY+RX×IY))`	*IX* given its final value. Note that 2nd closing parenthesis encloses those functions that are multiplied by 0.
`-L(RX:G(R))`	Assigns *RX* its proper value.
`+MUL):`	End of execution when *MUL* is solved for. Multiplication by zero has been used to assign proper values to variables. This leaves the equation $-RX+MUL=0$. This is solved directly to obtain $MUL=RX$. Note the closing parenthesis after *MUL* to finish the IF(S(DIV) argument. The colon refers to the 2nd half of the IF(S(MUL) OR S(DIV) argument.
`IF(S(ADD):`	Execution jumps to here if neither *MUL* or *DIV* are desired and checks to see if *ADD* is desired.
`ADD-L(RX:RX+RY)`	If *ADD* is desired, *RX* is assigned its proper value.

`+0×L(IX:IX+IY):`	Assigns *IX* its proper value without disturbing the equation $ADD-RX=0$ that was generated in the previous line. This is solved directly to give $ADD=RX$.
`IF(S(SUB):`	If *ADD* is not desired execution jumps to here.
`SUB-L(RX:RX-RY)`	If *SUB* is desired, *RX* is assigned its proper value.
`+0×L(IX:IX-IY):`	Assigns *IX* its proper value without disturbing the equation $SUB-RX=0$ that was generated in the previous line. This is solved directly to give $SUB=RX$.
`IF(S(INV):`	Is $1/x$ desired?
`L(RX:RX÷L(R:SQ(RADIUS(RX:IX))))`	Assigns new value to *RX* while storing a^2+b^2 in intermediate variable *R* for use in next step.
`+0×L(IX:-IX÷G(R))`	Assigns new value to *IX*.
`-INV:`	Multiplication by zero leaves the effective equation $RX-INV=0$. This is solved directly to give $INV=RX$.
`0×(L(LNX:LN(RADIUS(RX:IX)))`	If *SUB* is not desired, execution jumps to here. *LNX* is used as an intermediate variable to store the final value of *RX* (old value of *RX* is still needed).
`+L(IX:ANGLE(RX:IX)))`	Final value of *IX* assigned if *LNX* is desired. If x^y is desired, *IX* will be changed again later. Note that the 3rd closing parenthesis encloses those functions multiplied by zero.

`-L(RX:G(LNX))`	Assigns *RX* its final value if *LNX* is desired. If x^y is desired, *RX* will be changed again later.
`+IF(S(LNX):`	Is *LNX* desired?
`LNX:`	If so, execution stops. Multiplication by zero has been used to assign proper values to variables. This leaves the equation $-RX+LNX=0$. This is solved directly to obtain $LNX=RX$.
`LNX+0×L(R:RX×RY-IX×IY)`	If *LNX* is not desired, then the only option left is *XY* (x^y). *LNX* is added to the -L(RX:G(LNX)) encountered two lines previously to give a zero result. Then, *R* is used as an intermediate variable to store the real part of $y\ln(x)$.
`+0×(L(IX:IX×RY+RX×IY)`	*IX* is assigned an intermediate value of the imaginary part of $y\ln(x)$.
`+L(RX:G(R)))`	*RX* is assigned an intermediate value since its old value is no longer needed. Note that the 3rd closing parenthesis encloses those functions that are multiplied by zero.
`-L(RX:EXP(G(R))×COS(IX))`	Final value of *RX* is assigned.
`+0×L(IX:SIN(IX)×EXP(G(R)))`	Final value of *IX* is assigned and multiplied by zero.
`+XY)))))`	The remaining equation is $-RX+XY=0$. This is solved directly to give $XY=RX$. The six closing parentheses are required to conclude all of the nested IFs.

Remarks on Using the Equation.

- All functions (*ADD*, *SUB*, *XY*, *LNX*, *MUL*, *SWAP*, *INV*, and *DIV*) return the real part of the result when evaluated. The real and imaginary parts of the result are placed in *RX* and *IX* respectively and can be viewed using the RCL key. The real and complex parts of *y* are not changed.
- For *XY* and *LNX* to give proper results, radians mode *must* be set.
- The equation will not allow *RX*, *IX*, *RY*, or *IY* to be unknowns. A "solution" will be returned, but it will not be correct.
- An infinite family of solutions for $\ln(x)$ exist of the form

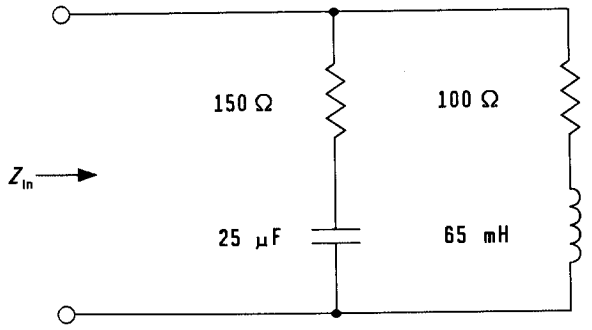
  $\ln(x)=\ln(r)+i(\theta+2\pi k)$, where $k=0,1,2,...$

  LNX returns the *principle value* of this family of solutions ($k=0$).
- *XY* likewise returns the principle value when an *n* th root is sought.

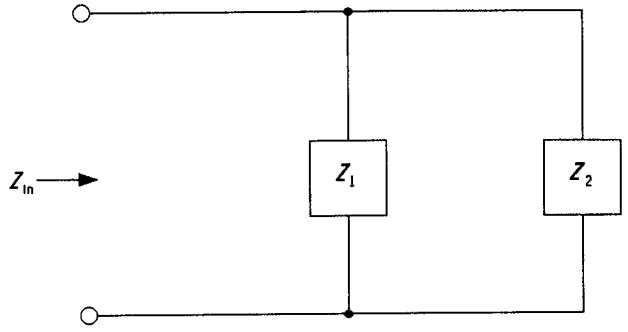
Example Problems

When your Solver equation matches the one listed, press CALC to display the menu of variables.

Example 1: Parallel Impedance. Find the total impedance Z_{in} for the circuit illustrated below. Assume the circuit is excited with a sinusoidal source of frequency f = 60 Hz.



The elements in series are simply added to give the branch impedances $Z_1 = 150 - i106.1033$ and $Z_2 = 100 + i24.5044$. The circuit is redrawn below.



The formula for two impedances in parallel is

$$Z_{in} = \frac{1}{\frac{1}{Z_1} + \frac{1}{Z_2}}$$

Keys:	Display:	Description:
150 [RX]	RX=150.0000	Stores real part of Z_1.
106.1033 [+/-] [IX]	IX=−106.1033	Stores imaginary part of Z_1.
100 [RY]	RY=100.0000	Stores real part of Z_2.
24.5044 [IY]	IY=24.5044	Stores imaginary part of Z_2.
[MORE] [INV]	INV=0.0044	Real part of $1/Z_1$.
[MORE] [MORE] [SWAP]	SWAP=100.0000	x $(1/Z_1)$ and y (Z_2) swapped. Real part of number in RX is displayed.
[MORE] [INV]	INV=0.0094	Calculates $1/Z_2$ and displays real part.
[ADD]	ADD=0.0139	Calculates $1/Z_1 + 1/Z_2$ and displays real part.
[INV]	INV=71.8042	Final result calculated and real part displayed.
[MORE] [MORE] [RCL] [IX]	IX=−4.3021	Recall imaginary part of final result.

Thus, the total impedance Z_{in} is 71.8042 − i4.3021 (Ω).

Example 2: Logarithm of a Negative Number. Find the natural logarithm of −12.

Be sure to set your calculator to radians mode before performing this example.

Keys:	Display:	Description:
■ [MODES] [MORE] [D/R] [EXIT] *		Set radians mode.
■ [CLEAR DATA] 12 [+/-] [RX]	RX=−12.0000	Sets all variables to zero and stores real part of x.

* To set radians mode on the HP-19B, use the following keystrokes: ■ [MODES] [D/R] [EXIT].

MORE		Real part of result.
MORE LNX	LNX=2.4849	
MORE		Imaginary part of result (π).
RCL IX	IX=3.1416	

Thus, $\ln(-12) = 2.4849 + i\pi$.

Example 3: Complex Roots. Find $\sqrt{27+i36}$.

Be sure to set your calculator to radians mode before performing this example.

Keys:	Display:	Description:
■ MODES MORE D/R EXIT *		Set radians mode, if necessary.
■ CLEAR DATA		Sets all variables to zero and stores real part of x.
27 RX	RX=27.0000	
36 IX	IX=36.0000	Stores IX.
.5 RY	RY=0.5000	Stores RY.
MORE MORE XY	XY=6.0000	Real part of $\sqrt{x}$.
MORE RCL IX	IX=3.0000	Imaginary part of $\sqrt{x}$.

Thus, $\sqrt{27+i36}=6+i3$.

* To set radians mode on the HP-19B, use the following keystrokes: ■ MODES D/R EXIT.

References

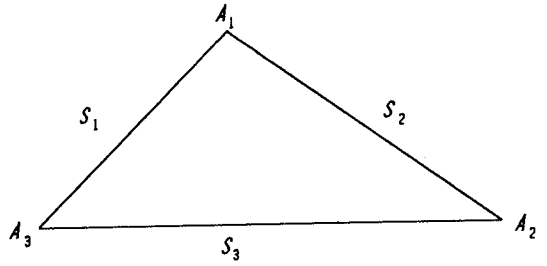
Boas, Mary L., *Mathematical Methods in the Physical Sciences*, John Wiley and Sons, New York, 1983.

Nilsson, James W., *Electric Circuits*, 2nd Edition, Addison-Wesley, Reading, MA, 1986.

Van Valkenburg, M. E., *Network Analysis*, 3rd Edition, Prentice-Hall, Englewood Cliffs, New Jersey, 1974.

Triangle Solutions

The triangle solutions Solver equation can be used to find the area, the lengths of the sides (S_1, S_2, S_3), and the angles (A_1, A_2, A_3) of a triangle.



Triangle Formulas

The following formulas are used as the basis for the triangle solutions Solver equation.

Side-Side-Side (S_1, S_2, S_3)

$$A_3 = 2\cos^{-1}\left[\frac{P(P-S_2)}{S_1 S_3}\right]^{\frac{1}{2}}$$

$$A_2 = 2\cos^{-1}\left[\frac{P(P-S_1)}{S_2 S_3}\right]^{\frac{1}{2}}$$

where $P = (S_1 + S_2 + S_3)/2$

$$A_1 = \cos^{-1}(-\cos(A_2 + A_3))$$

Angle-Side-Angle (A_3, S_1, A_1)

$$A_2 = \cos^{-1}(-\cos(A_1 + A_3))$$

$$S_2 = S_1 \frac{\sin A_3}{\sin A_2}$$

$$S_3 = S_1 \cos A_3 + S_2 \cos A_2$$

Side-Angle-Angle (S_1, A_1, A_2)

$$A_3 = \cos^{-1}(-\cos(A_1 + A_3))$$

The problem has been reduced to the A_3, S_1, A_1 configuration.

Side-Angle-Side (S_1, A_1, S_2)

$$S_3 = \sqrt{S_1^2 + S_2^2 - 2S_1 S_2 \cos A_1}$$

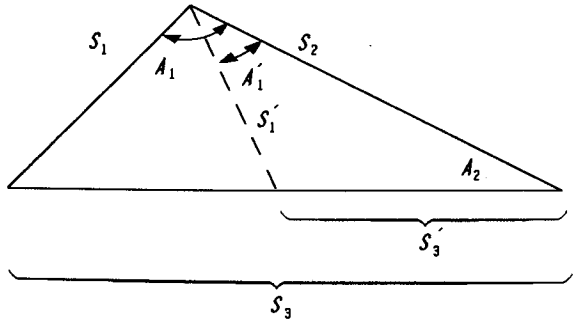
The problem has been reduced to the S_1, S_2, S_3 configuration.

Side-Side-Angle (S_1, S_2, A_2)

$$A_3 = \sin^{-1}\left[\frac{S_2}{S_1}\sin A_2\right]$$

$$A_1 = \cos^{-1}(-\cos(A_2 + A_3))$$

The problem has been reduced to the A_3, S_1, A_1 configuration. Note that two possible solutions exist for A_3 if A_3 is not equal to 90° and S_2 is greater than S_1, as shown below. Both possible answer sets are calculated.



In all cases, the area is calculated as

$$Area = \frac{1}{2} S_1 S_3 \sin A_3 .$$

The Triangle Solutions Equation

To use the Solver equation described below, simply key in the three known values and solve for the appropriate variable. After the calculator finishes solving the triangle, it displays the area. To view the sides and angles, use the RCL key and the appropriate menu key.

Equation:	Comments:
`TRIANGLE:`	Equation name.
`IF(S(SSS) OR S(SAS):`	Solving for *SSS* or *SAS*?
`IF(S(SAS):`	If so, solving for *SAS* specifically?
`0×L(S3:SQRT(SQ(S1)+SQ(S2)-2×S1×S2×COS(G(A1)))):`	Calculates *S* 3. Triangle is now in SSS configuration.
`0)`	Does nothing if *SSS* is sought.
`+0×L(P:(S1+S2+S3)÷2)`	Defines intermediate variable *P*.
`+0×L(A3:2×ACOS(SQRT(G(P)×(G(P)-S2)÷(S1×S3))))`	Calculates *A* 3 according to SSS formula.
`+0×L(A2:2×ACOS(SQRT(G(P)×(G(P)-S1)÷(S2×S3))))`	Calculates *A* 2 according to SSS formula.
`+0×L(A1:ACOS(-COS(0×A1+A2+A3)))`	Calculates *A* 1 according to SSS formula. Notice multiplication by zero employed to force *A* 1 to appear on the menu before *A* 2 and *A* 3.
`-.5×S1×S3×SIN(A3)`	Calculates area.

`+IF(S(SSS):SSS:SAS):`	Assigns area to proper variable. If solving for *SSS* or *SAS* execution stops here.
`IF(S(SSA):`	If not solving for *SSS* or *SAS*, execution jumps here. Checks to see if *SSA* is sought.
`IF(G(FLG)=1 AND S2>S1:`	If *SSA* is sought, check flag and side relationship to see which solution set will appear.
`0×L(A3:2×ASIN(1)-A3)+0×L(FLG:0):`	If flag is set, convert *A* 3 to its other possible value and clear the flag.
`0×L(FLG:1)+0×L(A3:ASIN(S2×SIN(A2)÷S1)))`	If flag is clear, calculate *A* 3 according to principle value of arcsine function. Set flag so the next time *SSA* is sought the other solution set will be generated.
`+0×L(A1:ACOS(-COS(A2+A3))):`	Calculates *A* 1. Problem is now reduced to ASA configuration.
`0)`	Other argument of IF(G(FLG)=1: from above. Does nothing if *SSA* is not sought.
`+IF(S(SAA):0×L(A3:ACOS(-COS(A1+A2))):`	If *SAA* is sought, calculate *A* 3 accordingly. Problem is now reduced to ASA configuration.
`0)`	Does nothing if *SAA* is not sought.
`+0×L(A2:ACOS(-COS(A1+A3)))`	Calculates *A* 2 according to ASA formula.
`+0×L(S2:S1×SIN(A3)÷SIN(A2))`	Calculates *S* 2 according to ASA formula.
`+0×L(S3:S1×COS(A3)+S2×COS(A2))`	Calculates *S* 3 according to ASA formula.

`-.5xS1xS3xSIN(A3)`	Calculates area.
`+IF(S(SSA):SSA:` `IF(S(SAA):SAA:ASA))`	Nested IF functions assign area to proper variable.

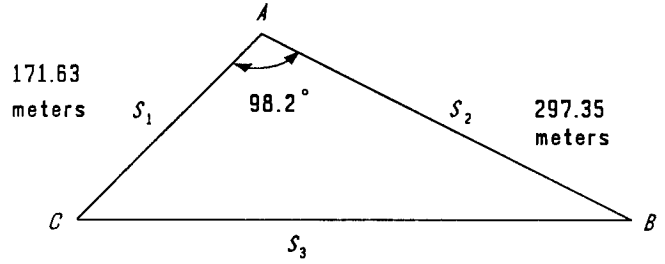
Remarks on Using the Equation

- Label the sides and angles of the triangle you wish to solve so they are consistent with the illustration on page 74. As you move clockwise around the triangle, the labels *must* be in the order $S_1, A_1, S_2, A_2, S_3, A_3$.
- Angles must be in units corresponding to the angular mode of the calculator (degrees or radians).
- Note that the triangle described by the equation does not conform to the standard triangle notation, i.e., A_1 is not opposite S_1.
- Angles must be entered as decimals. The ≡ >HR ≡ key can be used to convert degrees, minutes, and seconds to decimal degrees.
- The accuracy of a solution may degenerate for triangles containing extremely small angles.
- If the calculator displays `SOLUTION NOT FOUND`, there is no triangle satisfying the given dimensions.
- When the Solver equation uses the *SSA* formula, the first solution set is displayed. To see the second set, solve for *SSA* again (it is not necessary to re-enter the dimensions). Subsequent pushes of ≡ SSA ≡ will alternate between the two solution sets.

Example Problems.

Enter the Solver equation described above, taking care to put spaces around the OR functions and to use the proper number of parentheses. When you have finished, press ≡ CALC ≡ to display the menu of variables.

Example 1: Surveying a Land Parcel. A surveyor is to find the area and dimensions of a triangular land parcel. From point A, the distances to B and C are measured with an electronic distance meter. The angle between AB and AC is also measured. Find the area and other dimensions of the triangle.



This is a side-angle-side problem. Labeling the triangle to be consistent with the illustration on page 74 we have:

$$S_1 = 171.63,\ A_1 = 98.2°,\ \text{and}\ S_2 = 297.35.$$

For this example set the angular mode to degrees and the display format to "FIX 2."

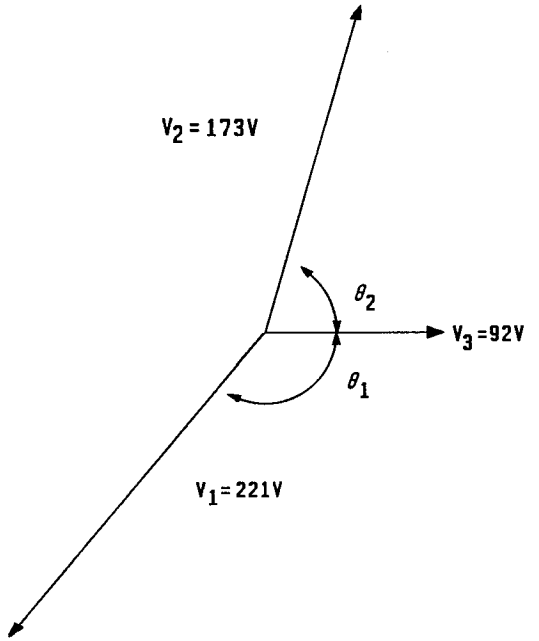
Keys:	Display:	Description:
■ MODES ≡ FIX ≡ 2 INPUT *		Set the display to FIX 2.
■ MODES ≡ MORE ≡ ≡ D/R ≡ EXIT †		Set degrees mode, if necessary.
171.63 ≡ S1 ≡	S1 = 171.63	Stores side 1.

* To set the display to FIX 2 on the HP-19B, use the following keystrokes: DISP ≡ FIX ≡ 2 INPUT.

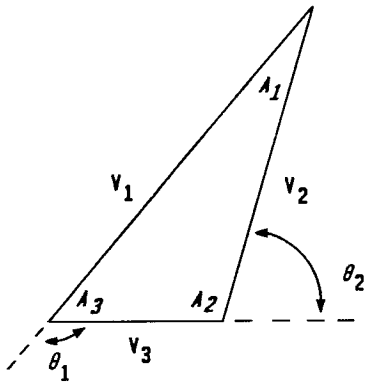
† To set degrees mode on the HP-19B, use the following keystrokes: ■ MODES ≡ D/R ≡ EXIT.

297.35 S2	S2=297.35	Stores side 2.
98.2 A1	A1=98.20	Stores angle 1.
MORE SAS	SAS=25,256.21	Area of land in square meters.
RCL A3	A3=53.97	Angle 3 (angle at point *C* in the figure).
MORE MORE RCL A2	A2=27.83	Angle 2 (angle at point *B* in the figure).
RCL S3	S3=363.91	Side 3 (*BC* in the figure).

Example 2: Phasor Diagrams. For a certain circuit it is known that the three voltages $V_1+V_2+V_3=0$. The voltages are complex quantities and can be expressed in a *phasor diagram* as shown below. The magnitudes are measured with a voltmeter and are as shown. Find the angular relationships between the voltages.



The phasor diagram can be re-drawn as the triangle shown below. We now have a side-side-side problem.



Make sure that the calculator is still in FIX 2 and degrees mode, as set in the previous example.

Keys:	Display:	Description:
221 S1	S1=221.00	Stores side 1.
173 S2	S2=173.00	Stores side 2.
92 S3	S3=92.00	Stores side 3.
MORE SSS	SSS=7,517.13	Area of triangle (no physical interpretation in this example).
RCL A3	A3=47.68	Angle 3.
MORE MORE RCL A1	A1=23.16	Angle 1.
RCL A2	A2=109.16	Angle 2.

Thus, $\theta_2 = 180° - A_2 = 70.84°$ and $\theta_2 = 180° - A_3 = 132.32°$.

Reference

Munem, M.A., Foulis, D.J., *College Trigonometry with Applications*, Worth Publishers, Inc., 1982

3 × 3 Matrix Operations

This Solver equation calculates both the determinant and, if it exists, the inverse of a 3 × 3 matrix. The inverse exists only if the determinant is not equal to zero. Using these results, a system of 3 linear equations in 3 unknowns may be solved.

Defining Equations

Let

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

then

$$A^{-1} = \frac{1}{\det} \begin{bmatrix} \begin{vmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{vmatrix} & - \begin{vmatrix} a_{12} & a_{13} \\ a_{32} & a_{33} \end{vmatrix} & \begin{vmatrix} a_{12} & a_{13} \\ a_{22} & a_{23} \end{vmatrix} \\ - \begin{vmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{vmatrix} & \begin{vmatrix} a_{11} & a_{13} \\ a_{31} & a_{33} \end{vmatrix} & - \begin{vmatrix} a_{11} & a_{13} \\ a_{21} & a_{23} \end{vmatrix} \\ \begin{vmatrix} a_{21} & a_{22} \\ a_{31} & a_{32} \end{vmatrix} & - \begin{vmatrix} a_{11} & a_{12} \\ a_{31} & a_{32} \end{vmatrix} & \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} \end{bmatrix}$$

where

$$\det = a_{11}(a_{22}a_{33} - a_{23}a_{32}) - a_{12}(a_{21}a_{33} - a_{23}a_{31}) + a_{13}(a_{21}a_{32} - a_{22}a_{31})$$

and

$$\begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc.$$

Solving a System of Linear Equations

A set of three linear equations in three unknowns can be written as:

$$a_{11}x_1 + a_{12}x_2 + a_{13}x_3 = c_1$$
$$a_{21}x_1 + a_{22}x_2 + a_{23}x_3 = c_2$$
$$a_{31}x_1 + a_{32}x_2 + a_{33}x_3 = c_3$$

where each a_{ij} is a constant coefficient, each c_i is a constant, and each x_j is an unknown.

This system of equations is written compactly in matrix form as

$$A\,\mathbf{x} = \mathbf{C}$$

where

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \quad \text{(coefficient matrix),}$$

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad \text{(unknown vector),}$$

and

$$\mathbf{C} = \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} \quad \text{(constant vector).}$$

The unknowns are then found as

$$\mathbf{x} = A^{-1}\mathbf{C}.$$

The 3 × 3 Matrix Operation Equation

This Solver equation uses the equations described previously to calculate the determinant and inverse of a 3 × 3 matrix and, using these results, to solve a set of three linear equations in three unknowns.

Equation:	Comments:
`MATRIX:`	Equation name.
`0×L(M:1÷(0×A11×A12+A13×(A21×G(A32)-A22×G(A31))-A12×(-A23×A31+0×A32+A21×A33)+A11×(A22×A33-A23×A32)))`	Assigns 1/det to intermediate variable M. Notice multiplication by zero used to arrange the variables $A11$ through $A33$ on the menu.
`×L(A:A22×A33-A23×A32)`	Assigns a_{11} of A^{-1} to intermediate variable A.
`×L(B:A12×A33-A13×A32)`	Assigns a_{12} of A^{-1} to intermediate variable B.
`×L(C:A12×A23-A13×A22)`	Assigns a_{13} of A^{-1} to intermediate variable C.
`×L(D:A21×A33-A23×A31)`	Assigns a_{21} of A^{-1} to intermediate variable D.
`×L(E:A11×A33-A13×A31)`	Assigns a_{22} of A^{-1} to intermediate variable E.
`×L(F:A11×A23-A13×A21)`	Assigns a_{23} of A^{-1} to intermediate variable F.
`×L(G:A21×A32-A22×A31)`	Assigns a_{31} of A^{-1} to intermediate variable G.
`×L(A32:G(M)×-(A11×A32-A12×A31))`	Assigns a_{32} of A^{-1} to $A32$.
`×L(A33:G(M)×(A11×A22-A12×A21))`	Assigns a_{33} of A^{-1} to $A33$.
`+0×L(A11:G(M)×G(A))`	Assigns a_{11} of A^{-1} to $A11$.
`×L(A12:G(M)×-G(B))`	Assigns a_{12} of A^{-1} to $A12$.
`×L(A13:G(M)×G(C))`	Assigns a_{13} of A^{-1} to $A13$.
`×L(A21:G(M)×-G(D))`	Assigns a_{21} of A^{-1} to $A21$.
`×L(A22:G(M)×G(E))`	Assigns a_{22} of A^{-1} to $A22$.
`×L(A23:G(M)×-G(F))`	Assigns a_{23} of A^{-1} to $A23$.
`×L(A31:G(M)×G(G))`	Assigns a_{31} of A^{-1} to $A31$.
`+IF(S(SIM):`	Solving for *SIM*?
`L(X1:0×X1×X2×X3+A11×C1+A12×C2+A13×C3)`	Uses multiplication by zero to arrange variables on the menu and assigns result to $X1$.
`+0×L(X2:A21×C1+A22×C2+A23×C3)`	Assigns result to $X2$.
`+0×L(X3:A31×C1+A32×C2+A33×C3)`	Assigns result to $X3$.
`-SIM:`	Generates equation $X1 - SIM = 0$, which is solved directly to give $SIM = X1$.
`1÷G(M)-DET)`	If *SIM* is not solved for, the equation $1/(1÷\text{det}) - DET = 0$ is solved directly to give $DET = \text{det}$.

Remarks on Using the Equation.

- When *DET* is calculated, the matrix elements are replaced by the elements of the inverted matrix. These new elements can be viewed by using the [RCL] key and the appropriate menu key.
- The Solver equation will return the value of $X1$ when *SIM* is calculated and will replace the matrix elements with the elements of the inverted matrix. $X2$, $X3$, and the inverted matrix elements can be viewed by using the [RCL] key and the appropriate menu key.
- A *homogeneous system* (every c_i equals zero) will return the trivial solution $\mathbf{x} = 0$.
- If the calculator displays `SOLUTION NOT FOUND` when solving for *DET* or *SIM*, the original matrix elements will remain intact. This message occurs when the computed value of $\det A = 0$. A matrix with a zero determinant is known as a *singular* matrix and has no inverse. When no inverse exists for the coefficient matrix in a system of simultaneous equations, either the system has no solution or infinitely many solutions. In some rare cases, the calculator may, through round-off error, return a zero value for $\det A$ when the actual value for the determinant is *not* zero. For matrices of this kind, an inverse *does* exist, but cannot be found using this Solver equation.

Example Problems

When your equation matches the matrix operation Solver equation, press [CALC] to display the menu of variables.

Example 1: Finding the Determinant and Inverse. Find the determinant and inverse of A where

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 1 & 3 & 3 \\ 1 & 2 & 4 \end{bmatrix}$$

Since the matrix has only integer elements, use the display format "ALL" for this example to eliminate extra trailing zeros.

Keys:	Display:	Description:
■ [MODES] * [ALL]		Set the display to ALL.
1 [A11]	A11 = 1	Stores A 11.
2 [A12]	A12 = 2	Stores A 12.
3 [A13]	A13 = 3	Stores A 13.
1 [A21]	A21 = 1	Stores A 21.
3 [A22]	A22 = 3	Stores A 22.
[MORE]		Stores A 23
3 [A23]	A23 = 3	
1 [A31]	A31 = 1	Stores A 31.
2 [A32]	A32 = 2	Stores A 32.
4 [A33]	A33 = 4	Stores A 33.
[MORE] [MORE]		Determinant of A.
[DET]	DET = 1	

* To set the display to ALL on the HP-19B, use the following keystrokes: [DISP] [ALL].

MORE		a_{11} of A^{-1}.
RCL A11	A11 = 6	
RCL A12	A12 = −2	a_{12} of A^{-1}.
RCL A13	A13 = −3	a_{13} of A^{-1}.
RCL A21	A21 = −1	a_{21} of A^{-1}.
RCL A22	A22 = 1	a_{22} of A^{-1}.
MORE		a_{23} of A^{-1}.
RCL A23	A23 = 0	
RCL A31	A31 = −1	a_{31} of A^{-1}.
RCL A32	A32 = 0	a_{32} of A^{-1}.
RCL A33	A33 = 1	a_{33} of A^{-1}.

Thus, $\det A = 1$, and

$$A^{-1} = \begin{bmatrix} 6 & -2 & -3 \\ -1 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix}$$

Example 2: A Singular Matrix. Find the inverse and determinant of A where

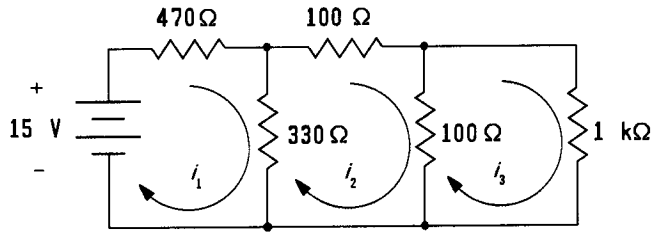
$$A = \begin{bmatrix} 4 & 6 & 8 \\ 5 & 7 & 5 \\ 2 & 3 & 4 \end{bmatrix}$$

As in the previous example, set the display format to "ALL."

Keys:	Display:	Description:
4 A11	A11 = 4	Stores A 11.
6 A12	A12 = 6	Stores A 12.
8 A13	A13 = 8	Stores A 13.
5 A21	A21 = 5	Stores A 21.
7 A22	A22 = 7	Stores A 22.
5 MORE A23	A23 = 5	Stores A 23.
2 A31	A31 = 2	Stores A 31.
3 A32	A32 = 3	Stores A 32.
4 A33	A33 = 4	Stores A 33.
MORE MORE DET	SOLUTION NOT FOUND	A^{-1} does not exist. Thus, A is singular and $\det A = 0$.

You may also wish to recall the elements of the original matrix to verify that they have not been altered.

Example 3: DC Circuit Analysis. For the circuit below, find i_1, i_2, and i_3. (i_1, i_2, and i_3 are all measured in mA.)



The following 3 equations are generated by summing the voltage drops around each loop.

$$\begin{aligned} 800i_1 - 330i_2 + 0i_3 &= 15 \\ -330i_1 + 530i_2 - 100i_3 &= 0 \\ 0i_1 - 100i_2 + 1100i_3 &= 0 \end{aligned}$$

Set your display format to "FIX 4" for this example.

Keys:	Display:	Description:
■ MODES FIX 4 INPUT *		Set the display to FIX 4, if necessary.
■ CLEAR DATA	0.0000	Sets all variables in equation equal to zero (eliminates having to explicitly store zeros in *A* 13, *A* 31, *C* 2, and *C* 3).
800 A11	A11 = 800.0000	Stores *A* 11.
330 +/- A12	A12 = −330.0000	Stores *A* 12.
STO A21	A21 = −330.0000	Stores *A* 21.
530 A22	A22 = 530.0000	Stores *A* 22.
MORE 100 +/- A23	A23 = −100.0000	Stores *A* 23.
STO A32	A32 = −100.0000	Stores *A* 32.
1100 A33	A33 = 1,100.0000	Stores *A* 33.
MORE 15 C1	C1 = 15.0000	Stores *C* 1.

* To set the display to FIX 4 on the HP-19B, use the following keystrokes: DISP FIX 4 INPUT.

MORE SIM	SIM = 0.0254	Solves for the 3 unknown currents and returns the value of i_1.
MORE MORE MORE		Recalls value of i_2.
RCL X2	X2 = 0.0161	
RCL X3	X3 = 0.0015	Recalls value of i_3.

Thus, i_1 = 25.4 mA, i_2 = 16.1 mA, and i_3 = 1.5 mA.

References

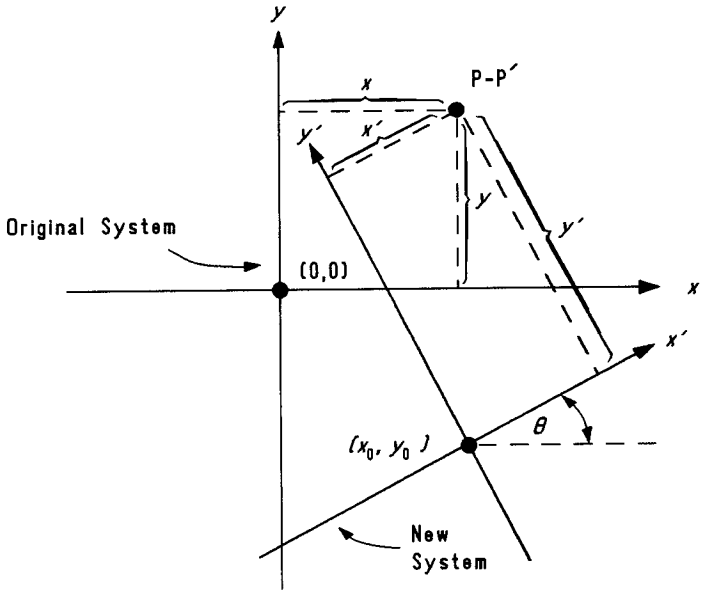
Florey, Francis G., *Elementary Linear Algebra with Applications*, Prentice-Hall, Englewood Cliffs, NJ, 1979.

Foulis, D. J. and M.A. Munem, *College Algebra with Applications*, Worth Publishers, Inc., 1982

Coordinate Transformations

The coordinate transformation Solver equation will provide two- and three-dimensional coordinate translation and/or rotation.

You must input the coordinates of the origin of the translated system (x_0, y_0, z_0), the rotation angle (θ) relative to the original system, and the axis about which rotation has occurred by giving a direction vector (ai, bj, ck) parallel to this axis. Note that the rotation axis passes through the translated origin (x_0, y_0, z_0). A point (x, y, z) in the original system can be converted to a point (x', y', z') in the new system. Inverse transformations are also possible. The figure below depicts two-dimensional translation and rotation.



Coordinate Transformation Formulas

The following formulas are used to develop the coordinate transformation Solver equation.

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} l_1 & m_1 & n_1 \\ l_2 & m_2 & n_2 \\ l_3 & m_3 & n_3 \end{bmatrix} \begin{bmatrix} x - x_0 \\ y - y_0 \\ z - z_0 \end{bmatrix}$$

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} l_1 & l_2 & l_3 \\ m_1 & m_2 & m_3 \\ n_1 & n_2 & n_3 \end{bmatrix} \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} + \begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix}$$

where

$$\begin{bmatrix} l_1 & l_2 & l_3 \\ m_1 & m_2 & m_3 \\ n_1 & n_2 & n_3 \end{bmatrix} = \begin{bmatrix} a^2(q)+\cos\theta & ab(q)-c\sin\theta & ac(q)+b\sin\theta \\ ba(q)+c\sin\theta & b^2(q)+\cos\theta & bc(q)-a\sin\theta \\ ca(q)-b\sin\theta & cb(q)+a\sin\theta & c^2(q)+\cos\theta \end{bmatrix}$$

and

$$q = (1-\cos\theta)$$

The Coordinate Transformation Equation

This Solver equation uses the matrix formulas described above to translate and/or rotate a two or three dimensional coordinate system.

Equation:	Comments:
CTRANS:	Equation name.
0xL(A:A÷L(D: SQRT(SQ(A)+SQ(B) +SQ(C)))	Changes a to a true direction vector component if it isn't already. The intermediate variable D is used for the magnitude of the original vector.

```
+0×L(B:B÷G(D))
```
Changes b to a true direction vector component if it isn't already.

```
+0×L(C:C÷G(D))
```
Changes c to a true direction vector component if it isn't already.

```
+0×L(L1:SQ(A)×L(Q:1-
COS(ANG))+COS(ANG))
```
Assigns proper value to $L1$. Note that Q serves the role of q in the defining equations.

```
+0×L(L2:A×B×G(Q)-
C×SIN(ANG))
```
Assigns proper value to $L2$.

```
+0×L(L3:A×C×G(Q)+
B×SIN(ANG))
```
Assigns proper value to $L3$.

```
+0×L(M1:G(L2)+2×C
×SIN(ANG))
```
Assigns proper value to $M1$.

```
+0×L(M2:SQ(B)×G(Q)+
COS(ANG))
```
Assigns proper value to $M2$.

```
+0×L(M3:B×C×G(Q)
-A×SIN(ANG))
```
Assigns proper value to $M3$.

```
+0×L(N1:G(L3)-2×B×
SIN(ANG))
```
Assigns proper value to $N1$.

```
+0×L(N2:G(M3)+2×A×
SIN(ANG))
```
Assigns proper value to $N2$.

```
+0×L(N3:SQ(C)×G(Q)+
COS(ANG))
```
Assigns proper value to $N3$.

```
+IF(S(FTRN):
```
Solving for *FTRN*?

```
L(X':G(L1)×(X-G(X0))
+G(M1)×(Y-G(Y0))+
G(N1)×(Z-G(Z0)))
```
Assigns proper result to X' during forward transformation.


```
+0×L(Y':G(L2)×(X
-X0)+G(M2)×(Y-Y0)+
G(N2)×(Z-Z0))
```
Assigns proper result to Y' during forward transformation.

```
+0×L(Z':G(L3)×(X
-X0)+G(M3)×(Y-Y0)+
G(N3)×(Z-Z0))
```
Assigns proper result to Z' during forward transformation.

```
-FTRN:
```
Generates the equation $X'-FTRN=0$, which is solved directly to give $FTRN=X'$.

```
L(X:G(L1)×X'+G(L2)
×Y'+G(L3)×Z'+X0)
```
Execution jumps to here when *not* solving for *FTRN* (implies that *ITRN* is sought). Assigns proper result to X when reverse transforming.

```
+0×L(Y:G(M1)×X'+
G(M2)×Y'+G(M3)×Z'+
Y0)
```
Assigns proper result to Y when reverse transforming.

```
+0×L(Z:G(N1)×X'+
G(N2)×Y'+G(N3)×Z'+
Z0)
```
Assigns proper result to Z when reverse transforming.

```
-ITRN)
```
Generates the equation $X-ITRN=0$, which is solved directly to give $ITRN=X$.

Remarks on Using the Equation.

- The sign of the rotation angle θ is determined by the direction of the rotation axis and the right-hand rule. Thus, with the thumb of the right hand pointing in the direction of the rotation axis, the fingers curl in the *positive* direction of θ.
- θ is given the variable name *ANG* in the Solver equation. It may be in either radians or degrees, depending upon the angular mode of the calculator.
- It is not necessary to key in a true direction vector parallel to the axis of rotation. Any parallel vector, whether it has unit magnitude or not, can be used. The equation will automatically adjust (a, b, c) so that they constitute a true direction vector of magnitude 1.
- Two-dimensional transformations are handled as a special case of three-dimensional transformation with (a, b, c) set to (0,0,1). This causes rotation to occur about the z-axis.
- For pure translation, input 0 for θ.
- For pure rotation, input zeros for x_0, y_0, and z_0.
- For forward transforms (original to new) use *FTRN* (Forward TRaNsform). For inverse transforms (new to original) use *ITRN* (Inverse TRaNsform). *FTRN* and *ITRN* return X' and X respectively. The other coordinates of the transformed point can be viewed by using RCL and the appropriate menu key.

Example Problems

The following examples show you how to use the Solver equation to perform coordinate transformations. When your Solver equation matches the one listed, press CALC to display the menu of variables.

Example 1: A Two-Dimensional Transformation. A two-dimensional coordinate system with origin (0,0) is translated to (7,−4). After translation, a 27° rotation occurs. Convert the points $P_1(-9,7)$ and $P_2(6,8)$ to equivalent coordinates in the translated rotated system.

For this example make sure that the calculator is set to degrees mode.

Keys:	Display:	Description:
■ MODES MORE D/R EXIT *		Set degrees mode if necessary.
■ CLEAR DATA	0.0000	Sets all variables equal to zero.
1 C	C=1.0000	Stores rotation axis (0,0,1).
27 ANG	ANG=27.0000	Stores rotation angle.
MORE 7 X0	X0=7.0000	Stores x-coordinate of translated origin.
4 +/− Y0	Y0=−4.0000	Stores y-coordinate of translated origin.
7 Y	Y=7.0000	Stores y-coordinate of P_1.
MORE MORE 9 +/− X	X=−9.0000	Stores x-coordinate of P_1.
MORE MORE FTRN	FTRN=−9.2622	x-coordinate of P_1 in new coordinate system.

* To set degrees mode on the HP-19B, use the following keystrokes: ■ MODES D/R EXIT.

RCL Y'	Y' = 17.0649	y-coordinate of P_1 in new coordinate system.
MORE 6 X	X = 6.0000	Stores x-coordinate of P_2.
MORE 8 Y	Y = 8.0000	Stores y-coordinate of P_2.
MORE FTRN	FTRN = 4.5569	x-coordinate of P_2 in new coordinate system.
RCL Y'	Y' = 11.1461	y-coordinate of P_2 in new coordinate system.

Now, convert the point $P'_3(2.7, -3.6)$ to its equivalent coordinates in the original system.

Keys:	Display:	Description:
2.7 X'	X' = 2.7000	Stores x-coordinate of P'_3 in new system.
3.6 +/- Y'	Y' = −3.6000	Stores y-coordinate of P'_3 in new system.
ITRN	ITRN = 11.0401	x-coordinate of P'_3 in original system.
MORE MORE RCL Y	Y = −5.9818	y-coordinate of P'_3 in original system.

Notice that once the translated origin, rotation angle, and rotation axis have been stored, points can be both forward and inverse transformed without re-entering this data.

Example 2: A Three-Dimensional Transformation. A three-dimensional coordinate system is translated to (2.45, 4.00, 4.25). After translation, a 62.5° rotation occurs about the (0, −1, −1) axis. In the original system, a point had the coordinates (3.9, 2.1, 7.0). What are the coordinates of the point in the translated rotated system?

As in the previous example, set the calculator to degrees mode.

Keys:	Display:	Description:
■ CLEAR DATA	0.0000	Clears all variables in "CTRANS" equation.
1 +/- B	B = −1.0000	Stores b-component of rotation axis.
STO C	C = −1.0000	Stores c-component of rotation axis.
62.5 ANG	ANG = 62.5000	Stores rotation angle.
MORE 2.45 X0	X0 = 2.4500	Stores x-coordinate of translated origin.
4 Y0	Y0 = 4.0000	Stores y-coordinate of translated origin.
4.25 Z0	Z0 = 4.2500	Stores z-coordinate of translated origin.
2.1 Y	Y = 2.1000	Stores y-coordinate of point in original system.
7 Z	Z = 7.0000	Stores z-coordinate of point in original system.
MORE MORE 3.9 X	X = 3.9000	Stores x-coordinate of point in original system.

MORE MORE FTRN	FTRN=3.5861	x-coordinate of point in new system.
RCL Y'	Y'=0.2609	y-coordinate of point in new system.
RCL Z'	Z'=0.5891	z-coordinate of point in new system.

In the translated, rotated system above, a point has the coordinates (1, 1, 1). What are the corresponding coordinates in the original system?

Keys:	Display:	Description:
1 X'	X'=1.0000	Stores x-component of point in new system.
STO Y'	Y'=1.0000	Stores y-component of point in new system.
STO Z'	Z'=1.0000	Stores z-component of point in new system.
ITRN	ITRN=2.9117	x-component of point in original system.
MORE MORE RCL Y	Y=4.3728	y-component of point in original system.
RCL Z	Z=5.8772	z-component of point in original system.

References

CRC Handbook of Mathematical Sciences, 5th Edition, page 354, Edited by William H. Beyer, CRC Press, Inc., 1978

The VNR Concise Encyclopedia of Mathematics, page 533, Edited by W. Gallert, et.al., Van Nostrand Reinhold Co., 1975

Special thanks to Rene S. Julian for contributing the original version of this program to the HP-65 Users' Library.

More Step-by-Step Solutions for Your HP-27S or HP-19B Calculator

These additional books offer a variety of examples and keystroke procedures to help set up your calculations the way *you* need them.

Practical routines show you how to use the built-in menus to solve problems more effectively, while easy-to-follow instructions help you create personalized menus.

Real Estate, Banking, and Leasing (00017-90019)

- Use the TVM menu for real estate, banking, and leasing calculations.
- Calculate the annual percentage rate of a loan with fees.
- Calculate discounted, adjustable-rate, and bi-weekly mortgages.
- Develop menus for graduated-payment and wrap-around mortgages.
- Estimate monthly payments and mortgage insurance.
- Use menus to calculate Rule of 78s, add-on loans, constant payment loans, loans with odd first periods, and leases with multiple payments in advance.
- Work with a variety of methods to evaluate savings plans.

Business Finance and Accounting (00017-90020)

- Calculate break-even point, profits, and standard business ratios.
- Make investment decisions using payback period, net present value, and internal rate of return.
- Solve for variances and other cost-accounting variables.
- Develop menus to calculate the sample size for an inventory audit.
- Perform financial statement, production, and inventory analyses.
- Forecast sales and units to manufacture.

Marketing and Sales (00017-90021)

- Forecast sales using moving averages, exponential growth curves, and linear regression.
- Determine price, mark-up, and profit.
- Estimate the financial feasibility of a new product.
- Estimate the elasticity of demand.
- Build a "quote maker" for accurate, on-the-spot quotes.
- Base a customized menu on your company's commission scale to calculate your commission on a product.

Personal Investment and Tax Planning (00017-90022)

- Evaluate savings and IRA plans.
- Solve for funds available upon premature distribution from an IRA.
- Calculate basic mortgage components and the annual percentage rate of a loan.
- Evaluate your investment alternatives among life insurance, treasury bills, bonds, stocks, mutual funds, and limited partnerships.
- Calculate the Beta of your portfolio, estimate your stock price volatility, target your gains, hedge with call options, and estimate margin account gain or loss.
- Determine your tax and inflation break-even point.

How to Order...

To order a book your dealer does not carry, call toll-free 1-800-538-8787 and refer to call code P270. Master Card, Visa, and American Express cards are welcome. For countries outside the U.S., contact your local Hewlett-Packard sales office.

Step-by-Step Solutions for Your HP-27S or HP-19B Calculator

This book contains advanced Solver techniques and a variety of applications, equations, and keystrokes to provide solutions for the science and engineering professions.

- Advanced Solver Techniques
  - Two Powerful New Functions: LET and GET
  - Keystroke Reduction Techniques
  - Recursive Equations
  - Menu Variable Arrangement
  - Solutions for More Than One Variable at a Time
  - Techniques for Forcing Iteration
  - Indefinite Loop Simulation
  - Use of Trigonometric Functions
- Technical and Scientific Solutions
  - Greatest Common Divisor and Least Common Multiple
  - Numerical Integration
  - Numerical Differentiation
  - Factors and Primes
  - Vector Operations
  - Complex Number Operations
  - Triangle Solutions
  - 3 × 3 Matrix Operations
  - Coordinate Transformations

HEWLETT PACKARD

Reorder Number
00027-90044

00027-90053 English
Printed in Canada 11/88

0 88698 00020 5