Appendix A

Error Conditions

If you attempt a calculation containing an improper operation—say division by zero—the display will show **Error** and a number. To clear an error message, press any one key. This also restores the display prior to the **Error** display.

The HP-15C has the following error messages. (The description of **Error 2** includes a list of statistical formulas used.)

Error 0: Improper Mathematics Operation

Illegal argument to math routine:

[÷], where $x = 0$.

[y^x], where:

- out of Complex mode, $y < 0$ and x is noninteger;
- out of Complex mode, $y = 0$ and $x \leqslant 0$; or
- in Complex mode, $y = 0$ and $\mathrm{Re}(x) \leqslant 0$.

[$\sqrt{x}$], where, out of Complex mode, $x < 0$.

[1/x], where $x = 0$.

[LOG], where:

- out of Complex mode, $x \leqslant 0$; or
- in Complex mode, $x = 0$.

[LN], where:

- out of Complex mode, $x \leqslant 0$; or
- in Complex mode, $x = 0$.

[SIN^{-1}], where, out of Complex mode, $|x| > 1$.

[COS^{-1}], where, out of Complex mode, $|x| > 1$.

[STO] [÷], where $x = 0$.

[RCL] [÷], where the contents of the addressed register $= 0$.

[Δ%], where the value in the Y-register is 0.

[HYP^{-1}] [COS], where, out of Complex mode, $x < 1$.

[HYP^{-1}] [TAN], where, out of Complex mode, $|x| > 1$.

[C$_{y,x}$] or [P$_{y,x}$], where:

- x or y is noninteger;

- $x < 0$ or $y < 0$;
- $x > y$;
- x or $y \geqslant 10^{10}$.

Error 1: Improper Matrix Operation

Applying an operation other than a matrix operation to a matrix, that is, attempting a nonmatrix operation while a matrix is in the relevant register (whether the X- or Y-register or a storage register).

Error 2: Improper Statistics Operation

[x̄] $n = 0$
[s] $n \leqslant 1$
[ŷ,r] $n \leqslant 1$
[L.R.] $n \leqslant 1$

Error 2 is also displayed if division by zero or the square root of a negative number would be required during computation with any of the following formulas:

$$\bar{x} = \frac{\Sigma x}{n} \qquad \bar{y} = \frac{\Sigma y}{n}$$

$$s_x = \sqrt{\frac{M}{n(n-1)}} \qquad s_y = \sqrt{\frac{N}{n(n-1)}} \qquad r = \frac{P}{\sqrt{M \cdot N}}$$

$$A = \frac{P}{M} \qquad B = \frac{M\Sigma y - P\Sigma x}{n \cdot M}$$

$$\hat{y} = \frac{M\Sigma y + P(n \cdot x - \Sigma x)}{n \cdot M}$$

where:

$$M = n\Sigma x^2 - (\Sigma x)^2$$

$$N = n\Sigma y^2 - (\Sigma y)^2$$

$$P = n\Sigma xy - \Sigma x \Sigma y$$

(A and B are the values returned by the operation [L.R.], where $y = Ax + B$.)

Error 3: Improper Register Number or Matrix Element

Storage register named is nonexistent or matrix element indicated is nonexistent.

Error 4: Improper Line Number or Label Call

Line number called for is currently unoccupied or nonexistent (>448); or you have attempted to load a program line without available space; or the label called does not exist.

Error 5: Subroutine Level Too Deep

Subroutine nested more than seven deep.

Error 6: Improper Flag Number

Attempted a flag number >9.

Error 7: Recursive [SOLVE] or [∫]

A subroutine which is called by [SOLVE] also contains a [SOLVE] instruction; a subroutine which is called by [∫] also contains an [∫] instruction.

Error 8: No Root

[SOLVE] unable to find a root using given estimates.

Error 9: Service

Self-test discovered circuitry problem, or wrong key pressed during key test. Refer to appendix F.

Error 10: Insufficient Memory

There is not enough memory available to perform a given operation.

Error 11: Improper Matrix Argument

Inconsistent or improper matrix arguments for a given matrix operation:

[+] or [−], where the dimensions are incompatible.

[x], where:

- the dimensions are incompatible; or
- the result is one of the arguments.

[1/x], where the matrix is not square.

scalar/matrix [÷], where the matrix is not square.

[÷], where:

- the matrix in the X-register is not square;
- the dimensions are incompatible; or
- the result is the matrix in the X-register.

[MATRIX] 2, where the input is a scalar; or the number of rows is odd.

[MATRIX] 3, where the input is a scalar; or the number of columns is odd.

[MATRIX] 4, where the input is scalar.

[MATRIX] 5, where:

- the input is a scalar;
- the dimensions are incompatible; or
- the result is one of the arguments.

[MATRIX] 6, where:

- the input is scalar;
- the dimensions are incompatible (including the result); or
- the result is one of the arguments.

[MATRIX] 9, where the matrix is not square.

[RCL] [DIM] [I], where contents of R_I are scalar.

[DIM] [I], where contents of R_I are scalar.

[STO] [RESULT], where the input is scalar.

[$P_{y,x}$], where the number of columns is odd.

[$C_{y,x}$], where the number of rows is odd.

Pr Error (*Power Error*)

Continuous Memory interrupted and reset because of power failure.

Appendix B

Stack Lift and the LAST X Register

The HP-15C calculator has been designed to operate in a natural manner. As you have seen working through this handbook, most calculations do not require you to think about the operation of the automatic memory stack.

There are occasions, however—especially as you delve into programming—when you need to know the effect of a particular operation upon the stack. The following explanation should help you.

Digit Entry Termination

Most operations on the calculator, whether executed as instructions in a program or pressed from the keyboard, terminate digit entry. This means that the calculator knows that any digits you key in after any of these operations are part of a new number.

The only operations that do *not* terminate digit entry are the digit entry keys themselves:

[0] through [9]
[.]
[CHS]
[EEX]
[←]

Stack Lift

There are three types of operations on the calculator based on how they affect stack lift. These are stack-disabling operations, stack-enabling operations, and neutral operations.

When the calculator is in Complex mode, each operation affects both the real and imaginary stacks. The stack lift effects are the same. In addition, the number keyed into the display (real X-register) *after any operation except* [←] *or* [CLx] is accompanied by the placement of a zero in the imaginary X-register.

Disabling Operations

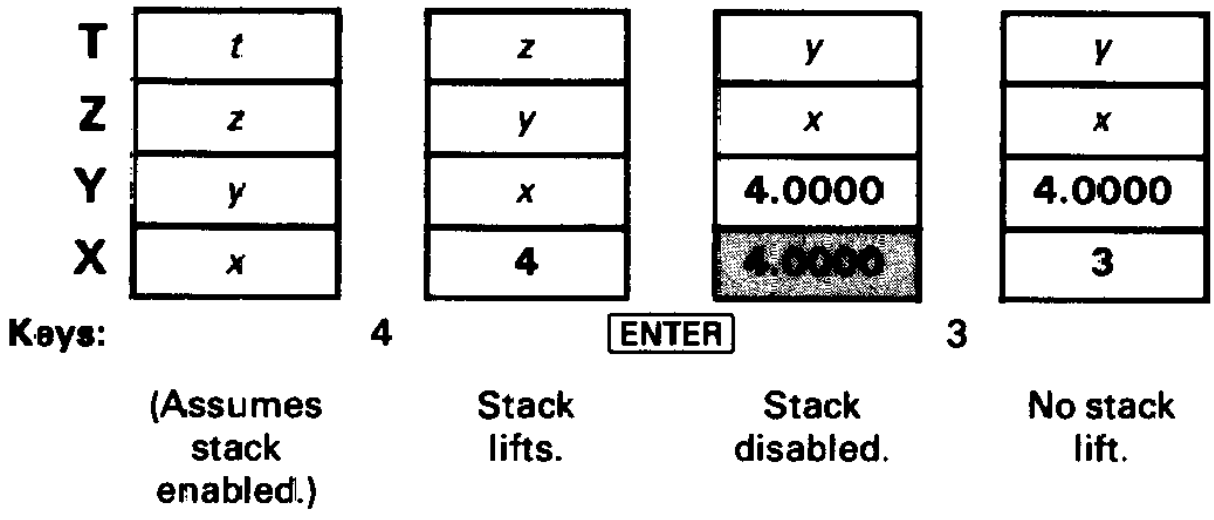
Stack Lift. There are four stack-disabling operations on the calculator.* These operations disable the stack lift, so that a number keyed in after one of these disabling operations writes over the current number in the displayed X-register and the stack does not lift. These special disabling operations are:

[ENTER] [CLx] [Σ+] [Σ-]

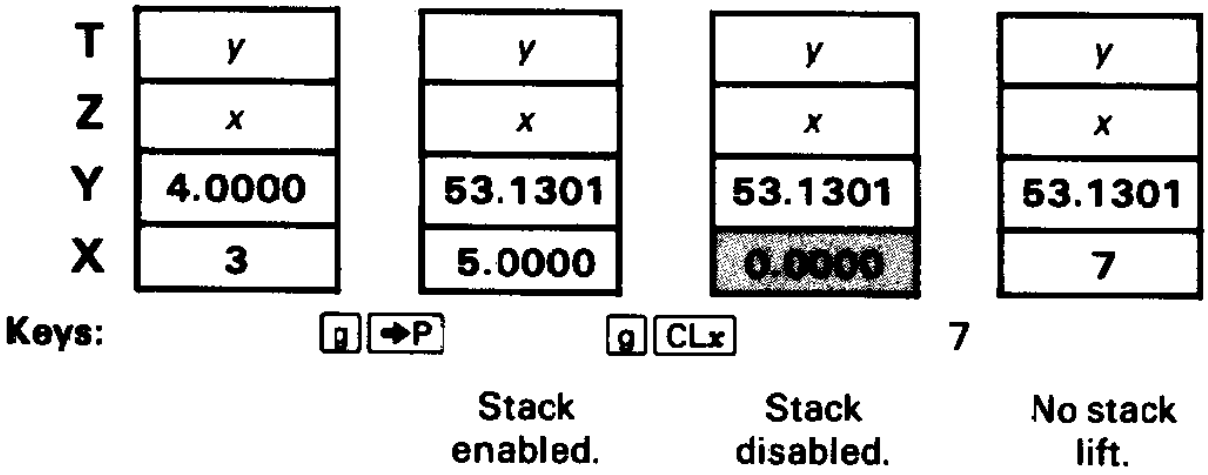
Imaginary X-Register. A zero is placed in the imaginary X-register when the next number following [ENTER], [Σ+], or [Σ-] is keyed or recalled into the display (real X-register). However, the next number keyed in or recalled after [←] or [CLx] *does not change* the contents of the imaginary X-register.

Enabling Operations

Stack Lift. Most of the operations on the keyboard, including one- and two-number mathematical functions like [x^2] and [×], are stack-enabling operations. This means that a number keyed in after one of these operations will lift the stack (because the stack has been "enabled" to lift). Both the real and imaginary stacks are affected. (Recall that a shaded X-register means that its contents will be written over when the next number is keyed in or recalled.)



* Refer to footnote, page 36.



Imaginary X-Register. All enabling functions provide for a zero to be placed in the imaginary X-register *when the next number is keyed or recalled into the display.*

Neutral Operations

Stack Lift. Some operations, like [FIX], are neutral; that is, they do not alter the previous status of the stack lift. Thus, if you disable the stack lift by pressing [ENTER], then press [f] [FIX] *n* and key in a new number, that number will write over the number in the X-register and the stack will not lift. Similarly, if you have previously enabled the stack lift by executing, say [√x], then execute a [FIX] instruction followed by a digit entry sequence, the stack will lift.*

The following operations are neutral on the HP-15C:

[FIX]	[GRD]	[USER]	[R/S]
[SCI]	[GTO] [CHS] *nnn*	CLEAR [PREFIX]	[P/R]
[ENG]	[BST]	CLEAR [REG]	[i]†
[DEG]	[SST]	CLEAR [Σ]	
[RAD]	[MEM]	[PSE]	.

Imaginary X-Register. The above operations are also neutral with respect to clearing the imaginary X-register.

---

* All digit entry functions are also neutral *during* digit entry. After digit entry termination, [CHS] and [EEX] are lift-enabling; [←] is disabling.

† That is, the [f] [i] sequence used to view the imaginary X-register.

LAST X Register

The following operations save x in the LAST X register:

[−]	[x^2]	[HYP^{-1}] [COS]	[%]
[+]	[SIN]	[HYP^{-1}] [TAN]	[Δ%]
[×]	[COS]	[→H.MS]	[→P]
[÷]	[TAN]	[→H]	[→R]
[ABS]	[SIN^{-1}]	[→DEG]	[$P_{y,x}$] *
[FRAC]	[COS^{-1}]	[→RAD]	[$C_{y,x}$] *
[INT]	[TAN^{-1}]	[LN]	[Σ+]
[RND]	[HYP] [SIN]	[e^x]	[Σ−]
[1/x]	[HYP] [COS]	[LOG]	[$\hat{y}$,r]
[$x!$]	[HYP] [TAN]	[10^x]	[MATRIX] 5 through 9
[$\sqrt{x}$]	[HYP^{-1}] [SIN]	[y^x]	[f] †

* Except when used as a matrix function.

† [f] uses the LAST X register in a special way, as described in appendix E.

Appendix C

Memory Allocation

The Memory Space

Storage registers, program lines, and advanced function execution* all draw on a common memory space in the HP-15C. The *availability* of memory for a specific purpose depends on the current *allocation* of memory, as well as on the total memory capacity of the calculator.

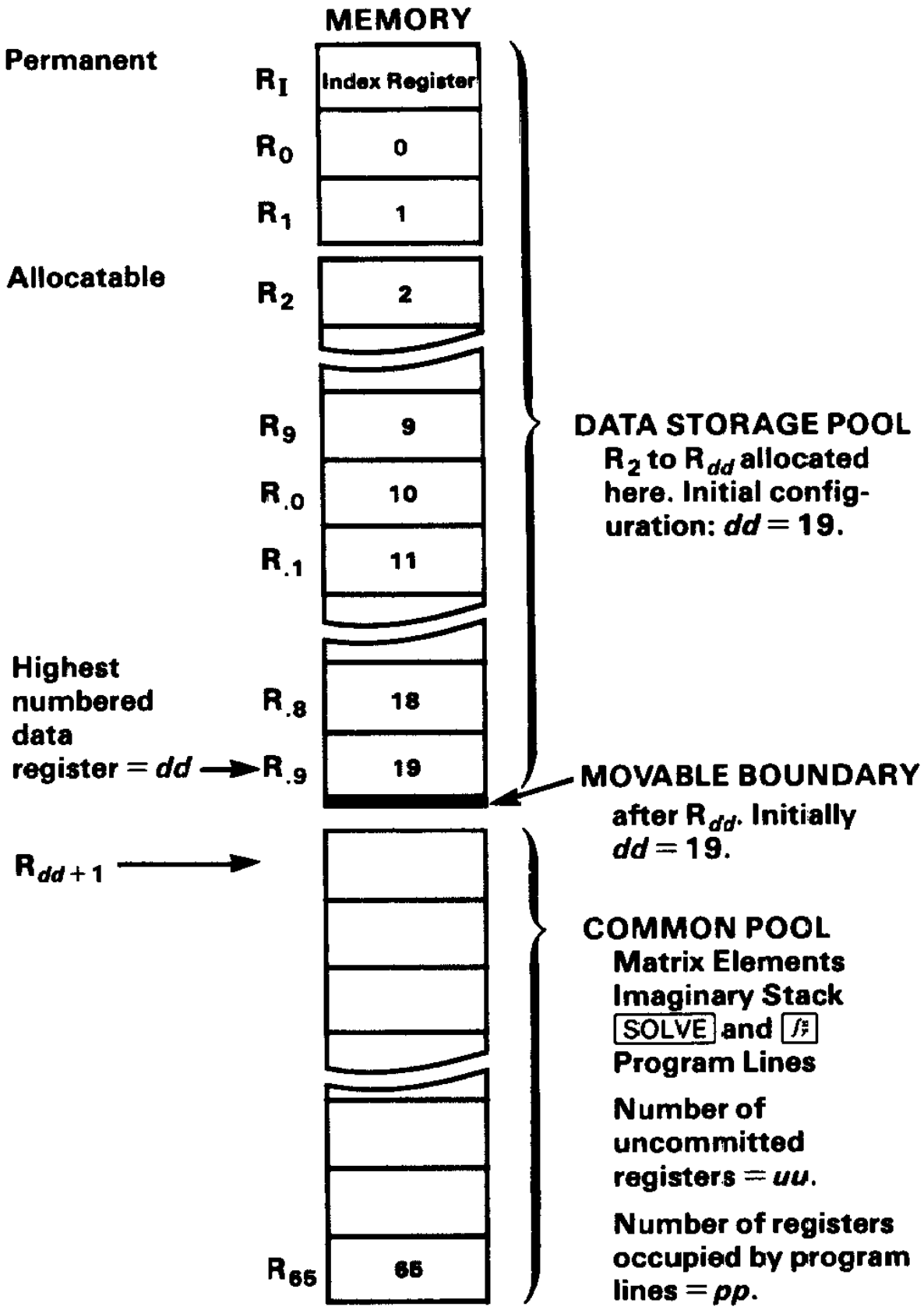
Registers

Memory space in the HP-15C is allocated on the basis of *registers.* This space is partitioned into two pools, which strictly define how a register may be used. There is always a combined total of 67 registers in these two pools.

- The *data storage pool* contains registers which may be used *only* for data storage. At power-up (Continuous Memory reset) this equals 21 registers. This pool contains at least three registers at all times: R_I, R_0, and R_1.

- The *common pool* contains uncommitted registers available for allocation to programming, matrices, the imaginary stack, and [SOLVE] and [∫] operation. At power-up there are 46 uncommitted registers in the common pool.

*The use of [SOLVE], [∫], Complex mode, or matrices temporarily requires extra memory space, as explained later in this appendix.



Total allocatable memory: 64 registers, numbered R_2 through R_{65}. $[(dd - 1) + uu + pp +$ (matrix elements) + (imaginary stack) + (SOLVE and f)$] = 64$. For memory allocation and indirect addressing, data registers $R_{.0}$ through $R_{.9}$ are referred to as R_{10} through R_{19}.

Memory Status ([MEM])

To view the current memory configuration of the calculator, press [g] [MEM] (*memory*), holding [MEM] to retain the display.* The display will be four numbers,

dd uu pp-b

where:

dd = the number of the *highest-numbered* register in the data storage pool (making the *total number* of data registers *dd* + 2 because of R_0 and R_I);

uu = the number of *uncommitted* registers in the common pool;

pp = the number of registers containing *program* instructions; and

b = the number of *bytes* left before *uu* is decremented (to supply seven more bytes of program memory) and *pp* is incremented.

The initial status of the HP-15C at power-up is:

19 46 0-0

The movable boundary between the data storage and common pools is always between R_{dd} and R_{dd+1}.

Memory Reallocation

There are 67 registers in memory, worth seven bytes each. Sixty-four of these registers (R_2 to R_{65}) are interconvertible between the data storage and common pools.

The [DIM] [(i)] Function

If you should require more common space (as for programming) or more data storage space (but not both simultaneously!), you can make the necessary register reallocation using [DIM] [(i)].† The procedure is:

---

*[MEM] is nonprogrammable.

†[DIM] (*dimension*) is so called because it is also used (with [A] through [E] or [I]) to dimension matrices. Above, however, it is used (with [(i)]) to "dimension" the size of the data storage pool.

1. Place *dd*, the *number of the highest data storage register you want allocated,* into the display. $1 \leqslant dd \leqslant 65$. The number of registers in the uncommitted pool (and therefore *potentially* available for programming) will be $(65 - dd)$.
2. Press [f] [DIM] [(i)].

There are two ways to review your allocation:

- Press [RCL] [DIM] [(i)] to recall into the stack the number of the highest-allocated data storage register, *dd*. (Programmable.)
- Press [g] [MEM] (as explained above) to view a more complete memory status (*dd uu pp-b*).

Keystrokes	Display	
(assuming a *cleared program memory*)*		
1 [f] [DIM] [(i)] [g] [MEM] (hold)	**1.0000** **1 64 0-0,**	R_1, R_0, and R_I allocated for data storage. Sixty-four registers are uncommitted; none contain program instructions.
19 [f] [DIM] [(i)] [RCL] [DIM] [(i)]	**19.0000** **19.0000**	R_{19} ($R_{.9}$) is the highest-numbered data storage register. Forty-six registers left in the common pool.

Restrictions on Reallocation

Continuous Memory will maintain the configuration you allocate until a new [DIM] [(i)] is executed or Continuous Memory is reset. If you try to allocate a number less than 1, *dd* = 1. If you try to allocate a number greater than 65, **Error 10** results.

---

*If program memory is not cleared, the number of uncommitted registers (*uu*) is less, owing to allocation of registers to program memory (*pp*). Therefore, *pp* would be > 0 and *b* would vary.

When converting registers, note that:

- You can convert registers from the common pool *only if they are uncommitted.* If, for example, you try to convert registers which contain program instructions, you will get an **Error 10** (insufficient memory).
- You *can* convert *occupied* registers from the data storage pool, *causing a loss of stored data.* An **Error 3** results if you try to address a "lost"—that is, nonexistent—register. Therefore, it is good practice to store data in the lowest-numbered registers first, as these are the last to be converted.

Program Memory

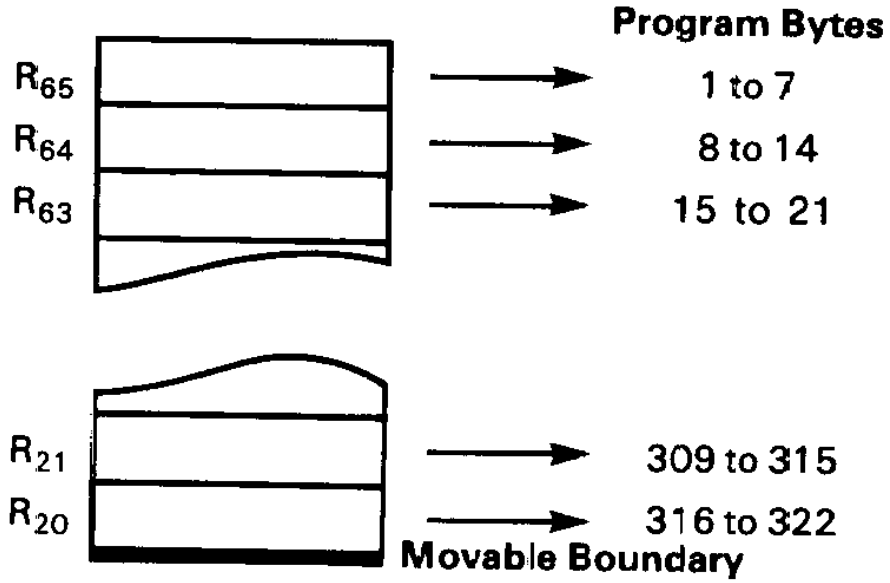
As mentioned before, each register consists of seven bytes of memory. Program instructions use one or two bytes of memory. Most program lines use one byte; those using two bytes are listed on page 218.

The *maximum* programming capacity of the HP-15C is 448 program bytes (64 convertible registers at seven bytes per register). At power-up, memory can hold up to 322 program bytes (46 allocated registers at seven bytes per register).

Automatic Program Memory Reallocation

Within the common register pool, program memory will automatically expand as needed. One uncommitted register at a time, starting with the highest-numbered register available, will be allocated to seven bytes of program memory.

Conversion of Uncommitted Registers to Program Memory



Your very first program instruction will commit R_{65} (all seven bytes) from an uncommitted register to a program register. Your eighth program instruction commits R_{64}, and so on, until the boundary of the common pool is encountered. Registers from the data storage pool (at power-up, this is R_{19} and below) are not available for program memory without reallocating registers using [DIM] [(i)].

Two-Byte Program Instructions

The following instructions are the only ones which require two bytes of calculator memory. (All others require only one byte.)

[f] [LBL] [·] *label*
[GTO] [·] *label*
[g] [CF] (*n* or [I])
[g] [SF] (*n* or [I])
[g] [F?] (*n* or [I])
[f] [FIX] (*n* or [I])
[f] [SCI] (*n* or [I])
[f] [ENG] (*n* or [I])
[f] [SOLVE]
[f] [∫]

[f] [MATRIX] {0 to 9}
[f] [x≷] {2 to 9, .0 to .9}
[f] [DSE] {2 to 9, .0 to .9}
[f] [ISG] {2 to 9, .0 to .9}
[STO] {[+], [−], [×], [÷]}
[RCL] {[+], [−], [×], [÷]}
[STO] [MATRIX] {[A] to [E]}
[STO] {[A] to [E], [(i)]} in User mode
[RCL] {[A] to [E], [(i)]} in User mode
[STO] [g] [(i)]
[RCL] [g] [(i)]

Memory Requirements for the Advanced Functions

The four advanced functions require temporary register space from the common register pool.

Function	Registers Needed
[SOLVE]	5
[∫]	23
	23 if executed together
Complex Stack	5
Matrices	1 per matrix element

For [SOLVE] and [∫], allocation and deallocation of the required register space takes place automatically.* Memory is thereby allocated only for the duration of these operations.

Space for the imaginary stack is allocated whenever [f] [I], [f] [Re≷Im], or [g] [SF] 8 is pressed. The imaginary stack is deallocated when [CF] 8 is executed.

Space for matrix registers is not allocated *until you dimension it* (using [DIM]). Reallocation takes place when you redimension a matrix. [MATRIX] 0 dimensions *all* matrices to 0×0.

*If you should interrupt a [SOLVE] or [∫] routine *in progress* by pressing a key, you could deallocate its registers by pressing [g] [RTN] or [f] CLEAR [PRGM] in Run mode.

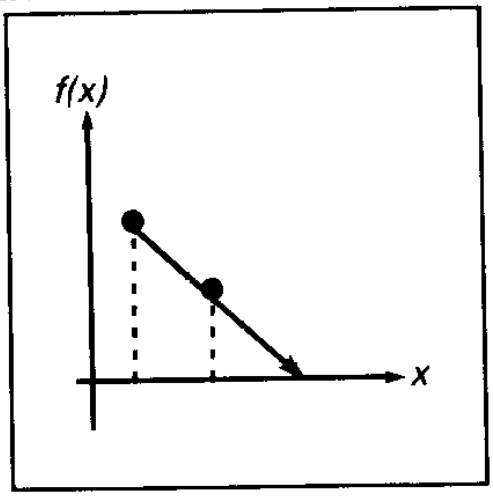
Appendix D

A Detailed Look at SOLVE

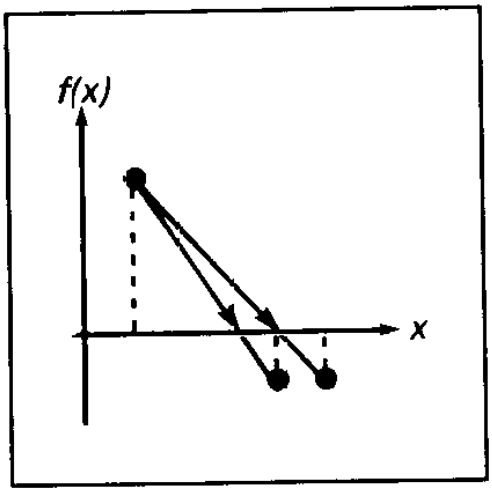
Section 13, Finding the Roots of an Equation, includes the basic information needed for the effective use of the SOLVE algorithm. This appendix presents more advanced, supplemental considerations regarding SOLVE.

How SOLVE Works

You will be able to use SOLVE most effectively by having a basic understanding of how the algorithm works.



In the process of searching for a zero of the specified function, the algorithm uses the value of the function at two or three previous estimates to approximate the shape of the function's graph. The algorithm uses this shape to intelligently "predict" a new estimate where the graph might cross the x-axis. The function subroutine is then executed, computing the value of the function at the new estimate. This procedure is performed repeatedly by the SOLVE algorithm.



If any two estimates yield function values with opposite signs, the algorithm presumes that the function's graph must cross the x-axis in at least one place in the interval between these estimates. The interval is systematically narrowed until a root of the equation is found.

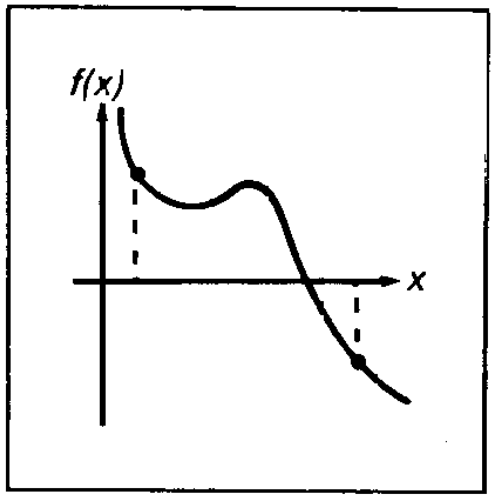
A root is successfully found either if the computed function value is equal to zero or if two estimates, differing by one unit in their last significant digit, give function values having opposite signs. In this case, execution stops and the estimate is displayed.

As discussed in section 13, page 186, the occurrence of other situations in the iteration process indicates the apparent absence of a function zero. The reason is that there is no way to logically predict a new estimate that is likely to have a function value closer to zero. In such cases, **Error 8** is displayed.

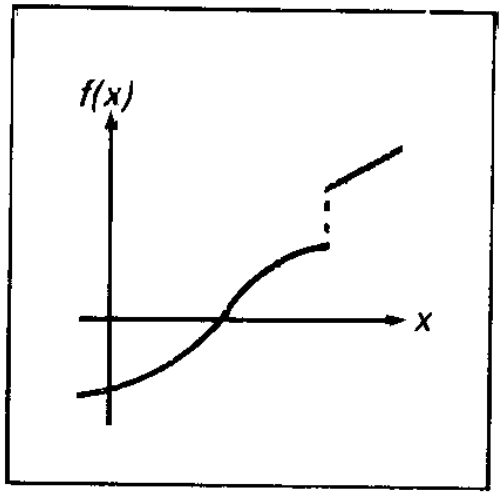
You should note that the initial estimates you provide are used to begin the "prediction" process. By permitting more accurate predictions than might otherwise occur, properly chosen estimates greatly facilitate the determination of the root you seek.

The SOLVE algorithm will *always* find a root provided one exists (within the overflow bounds), if any one of four conditions are met:

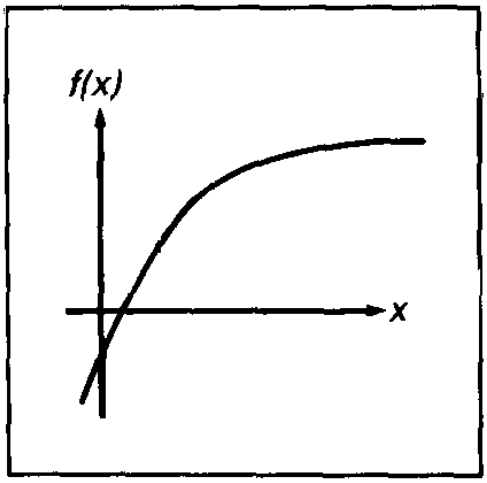
- Any two estimates have function values with opposite signs.



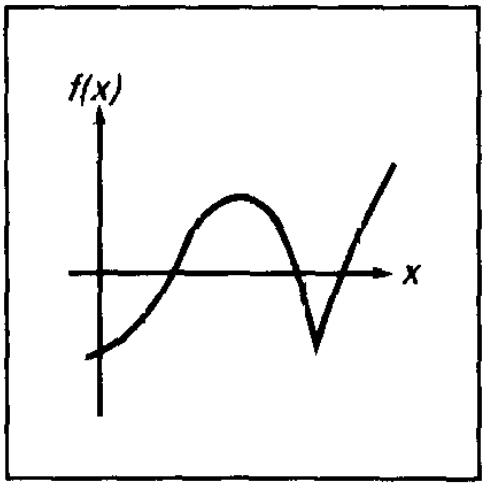
- The function is monotonic, meaning that $f(x)$ either always decreases or else always increases as x is increased.



- The function's graph is either convex everywhere or concave everywhere.



- The only local minima and maxima of the function's graph occur singly between adjacent zeros of the function.



In addition, it is assumed that the SOLVE algorithm will not be interrupted by an improper operation.

Accuracy of the Root

When you use the SOLVE key to find a root of an equation, the root is found accurately. The displayed root either gives a calculated function value ($f(x)$) exactly equal to zero or else is a 10-digit number virtually adjacent to the place where the function's graph crosses the x-axis. Any such root has an accuracy within two or three units in the 10th significant digit.

In most situations the calculated root is an accurate estimate of the theoretical (infinitely precise) root of the equation. However, certain conditions can cause the finite accuracy of the calculator to give a result that appears to be inconsistent with your theoretical expectation.

If a calculation has a result whose magnitude is smaller than $1.000000000 \times 10^{-99}$, the result is set equal to zero. This effect is referred to as "underflow." If the subroutine that calculates your function encounters underflow for a range of x and if this affects the value of the function, then a root in this range may be expected to have some inaccuracy. For example, the equation

$$x^4 = 0$$

has a root at $x = 0$. Because of underflow, SOLVE produces a root of **1.5060 -25** (for initial estimates of 1 and 2). As another example, consider the equation

$$1/x^2 = 0$$

whose root is infinite in value. Because of underflow, SOLVE gives a root of **3.1707 49** (for initial estimates of 10 and 20). In each of these examples, the algorithm has found a value of x for which the calculated function value equals zero. By understanding the effect of underflow, you can readily interpret results such as these.

The accuracy of a computed value sometimes can be adversely affected by "round-off" error, by which an infinitely precise number is rounded to 10 significant digits. If your subroutine requires extra precision to properly calculate the function for a range of x, the result obtained by SOLVE may be inaccurate. For example, the equation

$$|x^2 - 5| = 0$$

has a root at $x = \sqrt{5}$. Because no 10-digit number *exactly* equals $\sqrt{5}$, the result of using SOLVE is **Error 8** (for any initial estimates) because the function never equals zero nor changes sign. On the other hand, the equation

$$[(|x| + 1) + 10^{15}]^2 = 10^{30}$$

has no roots because the left side of the equation is always greater than the right side. However, because of round-off in the calculation of

$$f(x) = [(|x| + 1) + 10^{15}]^2 - 10^{30},$$

the root **1.0000** is found for initial estimates of 1 and 2. By recognizing situations in which round-off error may influence the operation of [SOLVE], you can evaluate the results accordingly and perhaps rewrite the function to reduce the effects of round-off.

In a variety of practical applications, the parameters in an equation—or perhaps the equation itself—are merely *approximations*. Physical parameters have an inherent accuracy (or inaccuracy). Mathematical representations of physical processes are only models of those processes, accurate only to the extent that the underlying assumptions are true. An awareness of these and other inaccuracies can be used to your advantage. By structuring your subroutine to return a function value of zero when the calculated value is negligible for practical purposes, you can usually save considerable time in finding a root with [SOLVE]—particularly for cases that would normally take a long time.

Example: Ridget hurlers such as Chuck Fahr can throw a ridget to heights of 105 meters and more. In fact, Fahr's hurls usually reach a height of 107 meters. How long does it take for his remarkable toss, described on page 184 in section 13, to reach 107 meters?

Solution: The desired solution is the value of t at which $h = 107$. Enter the subroutine from page 184 that calculates the height of the ridget. This subroutine can be used in a new function subroutine to calculate

$$f(t) = h(t) - 107.$$

The following subroutine calculates $f(t)$:

Keystrokes	Display	
[g][P/R]	000–	Program mode.
[f][LBL][B]	001–42,21,12	Begin with new label.
[GSB][A]	002– 32 11	Calculates $h(t)$.

Keystrokes	Display	
1	003– 1	
0	004– 0	
7	005– 7	Calculates $h(t) - 107$.
[−]	006– 30	
[g][RTN]	007– 43 32	

In order to find the first time at which the height is 107 meters, use initial estimates of 0 and 1 second and execute [SOLVE] using [B].

Keystrokes	Display	
[g] [P/R]		Run mode.
0 [ENTER]	0.0000	Initial estimates.
1	1	
[f] [SOLVE] [B]	4.1718	The desired root.
[R↓]	4.1718	A previous estimate of the root.
[R↓]	0.0000	Value of $f(t)$ at root.

It takes 4.1718 seconds for the ridget to reach a height of exactly 107 meters. (It takes approximately one minute to find this solution.)

However, suppose you assume that the function $h(t)$ is accurate only to the nearest whole meter. You can now change your subroutine to give $f(t) = 0$ whenever the calculated magnitude of $f(t)$ is less than 0.5 meter. Change your subroutine as follows:

Keystrokes	Display		
[g] [P/R]	000-		Program mode.
[GTO] [CHS] 006	006-	30	Line before [RTN] instruction.
[g] [ABS]	007-	43 16	Magnitude of $f(t)$.
[.]	008-	48	Accuracy.
5	009-	5	
[g] [TEST] 7	010-43,30,	7	Test for $x > y$ and return zero if accuracy > magnitude ($0.5 > \|f(t)\|$.
[g] [CLx]	011-	43 35	
[g] [TEST] 0	012-43,30,	0	Test for $x \neq 0$ and restore $f(t)$ if value is nonzero.
[g] [LSTx]	013-	43 36	

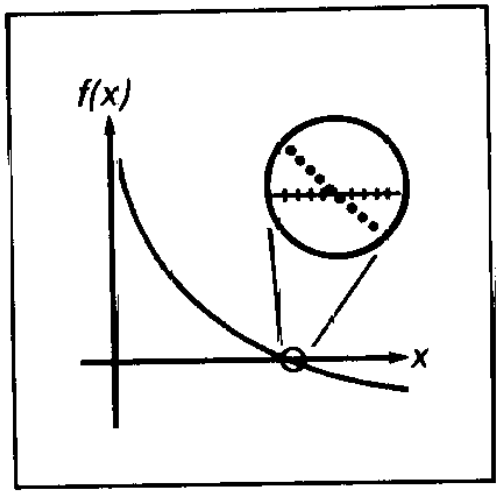
Execute [SOLVE] again:

Keystrokes	Display	
[g] [P/R]		Run mode.
0 [ENTER]	0.0000	Initial estimates.
1	1	
[f] [SOLVE] [B]	4.0681	The desired root.
[R↓]	4.0681	A previous estimate of the root.
[R↓]	0.0000	Value of modified $f(t)$ at root.

After 4.0681 seconds, the ridget is at a height of 107 ± 0.5 meters. This solution, although different from the previous answer, is correct considering the uncertainty of the height equation. (And this solution is found in just under half the time of the earlier solution.)

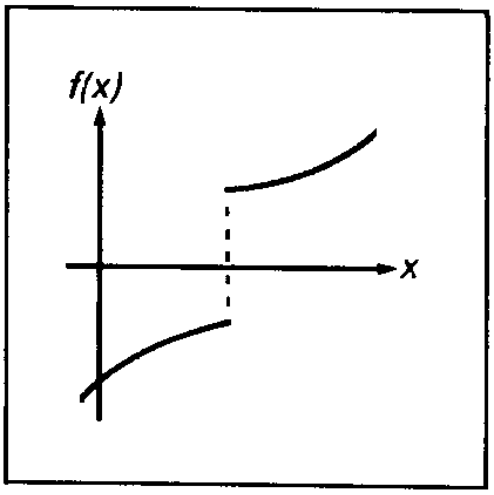
Interpreting Results

The numbers that [SOLVE] places in the X-, Y-, and Z-registers help you evaluate the results of the search for a root of your equation.* Even when no root is found, the results are still significant.

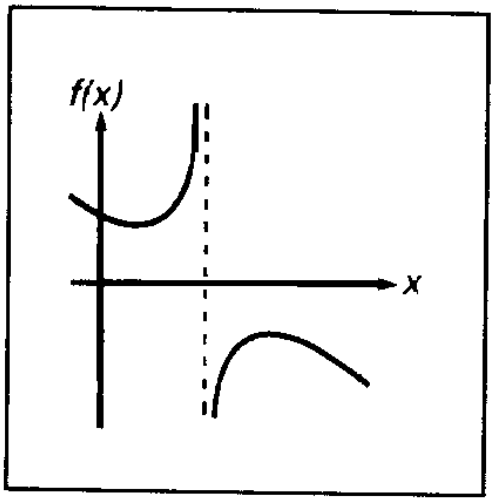


When [SOLVE] finds a root of the specified equation, the root and function values are placed in the X- and Z-registers. A function value of zero is the expected result. However, a nonzero function value is also acceptable because it indicates that the function's graph apparently crosses the x-axis within an infinitesimal distance from the calculated root. In most such cases, the function value will be relatively close to zero.

* The number in the T-register is the same number that was left in the Y-register by the final execution of your function subroutine. Generally, this number is not of interest.



Special consideration is required for a different type of situation in which SOLVE finds a root with a nonzero function value. If your function's graph has a discontinuity that crosses the x-axis, SOLVE specifies as a root an x-value adjacent to the discontinuity. This is reasonable because a large change in the function value between two adjacent values of x might be the result of a very rapid, continuous transition. Because this cannot be resolved by the algorithm, the root is displayed for you to interpret.



A function may have a *pole*, where its magnitude approaches infinity. If the function value changes sign at a pole, the corresponding value of x looks like a possible root of your equation, just as it would for any other discontinuity crossing the x-axis. However, for such functions, the function value placed into the Z-register when that root is found will be relatively large. If the pole occurs at a value of x that is *exactly* represented with 10 digits, the subroutine may try that value and halt prematurely with an error indication. In this case, the SOLVE operation will not be completed. Of course, this may be avoided by the prudent use of a conditional statement in your subroutine.

Example: In her analysis of the stresses in a structural component, design consultant Lucy I. Beame has determined that the shear stress can be expressed as

$$Q = \begin{cases} 3x^3 - 45x^2 + 350 & \text{for } 0 < x < 10 \\ 1000 & \text{for } 10 \leqslant x < 14 \end{cases}$$

where Q is the shear stress in newtons per square meter and x is the distance from one end in

meters. Write a subroutine to compute the shear stress for any value of x. Use [SOLVE] to find the location of zero shear stress.

Solution: The equation for the shear stress for x between 0 and 10 is more efficiently programmed after rewriting it using Horner's method:

$$Q = (3x - 45)x^2 + 350 \qquad \text{for } 0 < x < 10.$$

Keystrokes	Display		Comments
[g] [P/R]	000-		Program mode.
[f] [LBL] 2	001-42,21, 2		
1	002-	1	Test for x range.
0	003-	0	
[g] [$x \leqslant y$]	004-	43 10	
[GTO] 9	005-	22 9	Branch for $x \geqslant 10$.
[g] [CLx]	006-	43 35	
3	007-	3	
[×]	008-	20	$3x$.
4	009-	4	
5	010-	5	
[−]	011-	30	$(3x - 45)$.
[×]	012-	20	
[×]	013-	20	$(3x - 45)x^2$.
3	014-	3	
5	015-	5	
0	016-	0	
[+]	017-	40	$(3x - 45)x^2 + 350$.
[g] [RTN]	018-	43 32	End subroutine.
[f] [LBL] 9	019-42,21, 9		Subroutine for $x \geqslant 10$.
[EEX]	020-	26	
3	021-	3	$10^3 = 1000$.
[g] [RTN]	022-	43 32	End subroutine.

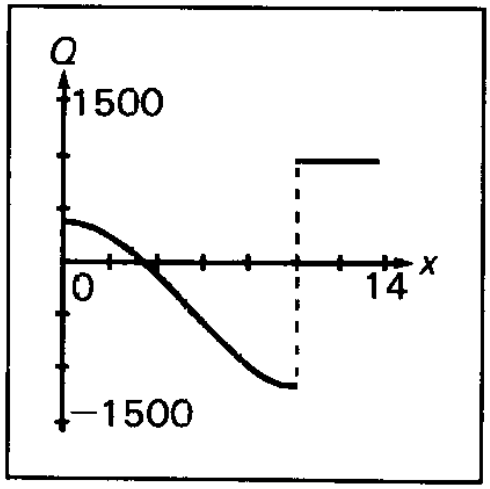
Execute [SOLVE] using initial estimates of 7 and 14 to start at the outer end of the beam and search for a point of zero shear stress.

Keystrokes	Display	
[g] [P/R]		Run mode.
7 [ENTER]	7.0000	Initial estimates.
14	14	
[f] [SOLVE] 2	10.0000	Possible root.
[R↓] [R↓]	1,000.0000	Stress not zero.

The large stress value at the root points out that the [SOLVE] routine has found a discontinuity. This is a place on the beam where the stress quickly changes from negative to positive. Start at the other end of the beam (estimates of 0 and 7) and use [SOLVE] again.

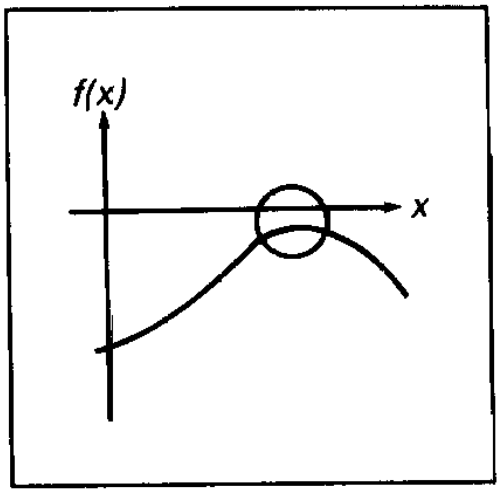
Keystrokes	Display	
0 [ENTER]	0.0000	Initial estimates.
7	7	
[f] [SOLVE] 2	3.1358	Possible root.
[R↓] [R↓]	2.0000 −07	Negligible stress.

Beame's beam has zero shear stress at approximately 3.1358 meters and an abrupt change of stress at 10.0000 meters.



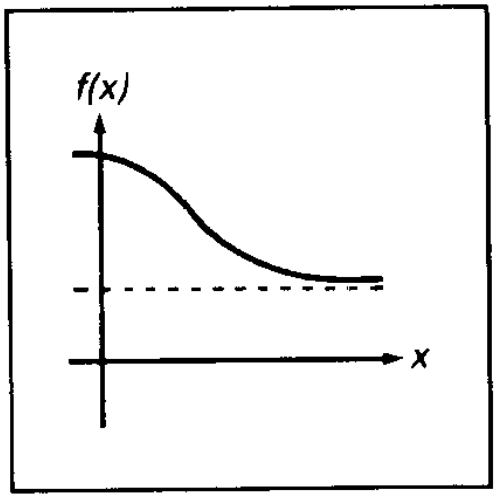
Graph of *Q* versus *x*.

When no root is found and **Error 8** is displayed, you can press [←] or any one key to clear the display and observe the estimate at which the function was closest to zero. By also reviewing the numbers in the Y- and Z-registers, you can often determine the nature of the function near the root estimate and use this information constructively.

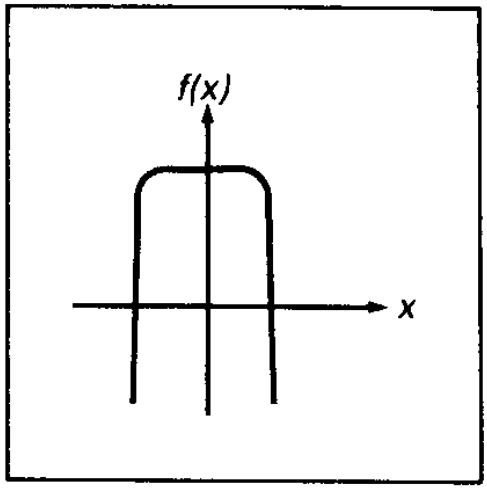


If the algorithm terminates its search near a local minimum of the function's *magnitude*, clear the **Error 8** display and observe the numbers in the X-, Y-, and Z-registers by rolling down the stack. If the value of the function saved in the Z-register is relatively close to zero, it is possible that a root of your equation has been found—the number returned in the X-register may be a 10-digit number very close to a theoretical root. You can explore this potential minimum further by rolling the stack until the returned estimates are back in the X- and Y-registers and then executing [SOLVE] again using these numbers as initial estimates. If an actual minimum has been found, **Error 8** will again be displayed and the number in the X-register will be approximately the same as before, but possibly closer to the actual location of the minimum.

Of course, you may deliberately use [SOLVE] to find the location of a local minimum of the function's magnitude. However, in this case you must be careful to confine the search in the region of the minimum. Remember, [SOLVE] tries hard to find a *zero* of the function.



If the algorithm stops searching and displays **Error 8** because it is working on a horizontal asymptote (when the value of the function is essentially constant for a large range of x), the estimates in the X- and Y-registers usually are significantly different from each other. The number in the Z-register is the value of the potential asymptote. If you execute [SOLVE] again using as initial estimates the numbers that were returned in the X- and Y-registers, a horizontal asymptote may again cause **Error 8**, but with numbers in the X- and Y-registers that will differ from the previous numbers. The value of the function in the Z-register would then be about the same as that obtained previously.



If **Error 8** is displayed as a result of a search that is concentrated in a local "flat" region of the function, the estimates in the X- and Y-registers will be relatively close together or extremely small. Execute [SOLVE] again using for initial estimates the numbers from the X- and Y-registers (or perhaps two numbers somewhat further apart). If the magnitude of the function is neither a minimum nor constant, the algorithm will eventually expand its search and find a more significant result.

Example: Investigate the behavior of the function

$$f(x) = 3 + e^{-|x|/10} - 2e^{x^2 e^{-|x|}}$$

as evaluated in the following subroutine.

Keystrokes	Display		
[g] [P/R]	000-		Program mode.
[f] [LBL] .0	001-42,21,.0		
[g] [ABS]	002-	43 16	
[CHS]	003-	16	
[e^x]	004-	12	$e^{-\|x\|}$.
[$x \gtrless y$]	005-	34	Bring x-value into X-register.
[g] [x^2]	006-	43 11	
[×]	007-	20	$x^2 e^{-\|x\|}$.
[e^x]	008-	12	
2	009-	2	
[×]	010-	20	
[CHS]	011-	16	$-2e^{x^2 e^{-\|x\|}}$.
[$x \gtrless y$]	012-	34	Bring x-value into X-register.
[g] [ABS]	013-	43 16	
[CHS]	014-	16	
1	015-	1	
0	016-	0	
[÷]	017-	10	$-\|x\|/10$.
[e^x]	018-	12	

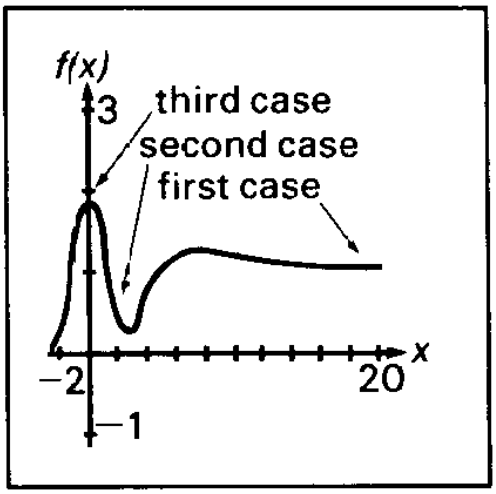
Keystrokes	Display		
[+]	019–	40	$e^{-\lvert x\rvert/10}-2e^{x^2e^{-\lvert x\rvert}}$.
3	020–	3	
[+]	021–	40	$3+e^{-\lvert x\rvert/10}-2e^{x^2e^{-\lvert x\rvert}}$.
[g] [RTN]	022–	43 32	

Use [SOLVE] with the following *single* initial estimates: 10, 1, and 10^{-20}.

Keystrokes	Display		
[g] [P/R]			Run mode.
10 [ENTER]	10.0000		Single estimate.
[f] [SOLVE] .0	Error 8		
[←]	455.4335		Best x-value.
[R↓]	48,026,721.85		Previous value.
[R↓]	1.0000		Function value.
[g] [R↑] [g] [R↑]	455.4335		Restore the stack.
[f] [SOLVE] .0	Error 8		
[←]	48,026,721.85		Another x-value.
[R↓] [R↓]	1.0000		Same function value (an asymptote).
1 [ENTER]	1.0000		Single estimate.
[f] [SOLVE] .0	Error 8		
[←]	2.1213		Best x-value.
[R↓]	2.1471		Previous value.
[R↓]	0.3788		Function value.
[g] [R↑] [g] [R↑]	2.1213		Restore the stack.
[f] [SOLVE] .0	Error 8		
[←]	2.1213		Same x-value.
[R↓] [R↓]	0.3788		Same function value (a minimum).
[EEX] [CHS] 20 [ENTER]	1.0000	–20	Single estimate.
[f] [SOLVE] .0	Error 8		
[←]	1.0000	–20	Best x-value.
[R↓]	1.1250	–20	Previous value.
[R↓]	2.0000		Function value.

Keystrokes	Display		
[g][R↑][g][R↑]	1.0000	-20	Restore the stack.
[f][SOLVE].0	Error 8		
[←]	1.1250	-20	Another x-value.
[R↓]	1.5626	-16	Previous value.
[R↓]	2.0000		Same function value.

In each of the three cases, [SOLVE] initially searched for a root in a direction suggested by the graph around the initial estimate. Using 10 as the initial estimate, [SOLVE] found the horizontal asymptote (value of 1.0000). Using 1 as the initial estimate, a minimum of 0.3788 at $x = 2.1213$ was found. Using 10^{-20} as the initial estimate, the function was essentially constant (at a value of 2.0000) for the small range of x that was sampled.



Finding Several Roots

Many equations that you encounter have more than one root. For this reason, you will find it helpful to understand some techniques for finding several roots of an equation.

The simplest method for finding several roots is to direct the root search in different ranges of x where roots may exist. Your initial estimates specify the range that is initially searched. This method was used for all examples in section 13. You can often find the roots of an equation in this manner.

Another method is known as *deflation*. Deflation is a method by which roots are "eliminated" from an equation. This involves modifying the equation so that the first roots found are no longer roots, but the rest of the roots remain roots.

If a function $f(x)$ has a value of zero at $x = a$, then the new function $f(x)/(x - a)$ will not approach zero in this region (if a is a simple root of $f(x) = 0$). You can use this information to eliminate a known root. Simply add a few program lines at the end of your function subroutine. These lines should subtract the known root (to 10

significant digits) from the x-value and divide this difference into the function value. In many cases the root will be a simple one, and the new function will direct [SOLVE] away from the known root.

On the other hand, the root may be a *multiple root*. A multiple root is one that appears to be present repeatedly, in the following sense: at such a root, not only does the graph of $f(x)$ cross the x-axis, but its slope (and perhaps the next few higher-order derivatives) also equals zero. If the known root of your equation is a multiple root, the root is not eliminated by merely dividing by the factor described above. For example, the equation

$$f(x) = x(x-a)^3 = 0$$

has a multiple root at $x = a$ (with a multiplicity of 3). This root is not eliminated by dividing $f(x)$ by $(x-a)$. But it can be eliminated by dividing by $(x-a)^3$.

Example: Use deflation to help find the roots of

$$60x^4 - 944x^3 + 3003x^2 + 6171x - 2890 = 0.$$

Using Horner's method, this equation can be rewritten in the form

$$(((60x - 944)x + 3003)x + 6171)x - 2890 = 0.$$

Program a subroutine that evaluates the polynomial.

Keystrokes	Display	
[g] [P/R]	000-	Program mode.
[f] CLEAR [PRGM]	000-	
[f] [LBL] 2	001-42,21, 2	
6	002- 6	
0	003- 0	
[×]	004- 20	
9	005- 9	
4	006- 4	
4	007- 4	
[-]	008- 30	
[×]	009- 20	
3	010- 3	
0	011- 0	

Keystrokes	Display	
0	012–	0
3	013–	3
[+]	014–	40
[×]	015–	20
6	016–	6
1	017–	1
7	018–	7
1	019–	1
[+]	020–	40
[×]	021–	20
2	022–	2
8	023–	8
9	024–	9
0	025–	0
[−]	026–	30
[g] [RTN]	027–	43 32

In Run mode, key in two large, negative initial estimates (such as −10 and −20) and use [SOLVE] to find the most negative root.

Keystrokes	Display	
[g] [P/R]		Run mode.
10 [CHS] [ENTER]	−10.0000	Initial estimates.
20 [CHS]	−20	
[f] [SOLVE] 2	−1.6667	First root.
[STO] 0	−1.6667	Stores root for deflation.
[R↓] [R↓]	4.0000 −06	Function value near zero.

Return to Program mode and add instructions to your subroutine to eliminate the root just found.

Keystrokes	Display		
[g] [P/R]	000–		Program mode.
[g] [BST] [g] [BST]	026–	30	Line before [RTN].
[x⇄y]	027–	34	Brings x into X-register.
[RCL] 0	028–	45 0	Divides by $(x - a)$, where a is known root.
[−]	029–	30	
[÷]	030–	10	

Now use the same initial estimates to find the next root.

Keystrokes	Display	
[g] [P/R]	4.0000 -06	Run mode.
10 [CHS] [ENTER]	-10.0000	Same initial estimates.
20 [CHS]	-20	
[f] [SOLVE] 2	0.4000	Second root.
[STO] 1	0.4000	Stores root for deflation.
[R↓] [R↓]	0.0000	Deflated function value.

Now modify your subroutine to eliminate the second root.

Keystrokes	Display		Deflation for second root.
[g] [P/R]	000-		Program mode.
[g] [BST] [g] [BST]	030-	10	Line before [RTN].
[x≷y]	031-	34	Brings x into X-register.
[RCL] 1	032-	45 1	
[−]	033-	30	
[÷]	034-	10	

Again, use the same initial estimates to find the next root.

Keystrokes	Display	
[g] [P/R]	0.0000	Run mode.
10 [CHS] [ENTER]	-10.0000	Same initial estimates.
20 [CHS]	-20	
[f] [SOLVE] 2	8.4999	Third root.
[STO] 2	8.4999	Stores root for deflation.
[R↓] [R↓]	-1.0929 -07	Deflated function value near zero.

Now change your subroutine to eliminate the third root.

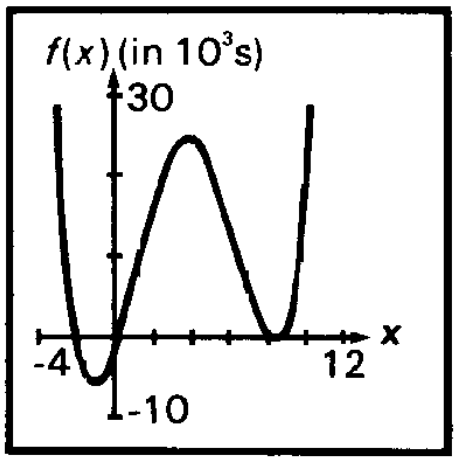
Keystrokes	Display		
[g] [P/R]	000-		Program mode.
[g] [BST] [g] [BST]	034-	10	Line before [RTN].
[x≷y]	035-	34	Brings x into X-register.

Keystrokes	Display	
[RCL] 2	036 45 2	} Deflation for third root.
[−]	037- 30	
[÷]	038- 10	

Find the fourth root.

Keystrokes	Display	
[g] [P/R]	-1.0929 -07	
10 [CHS] [ENTER]	-10.0000	} Same initial estimates.
20 [CHS]	-20	
[f] [SOLVE] 2	8.5001	Fourth root.
[STO] 3	8.5001	Stores root for reference.
[R↓] [R↓]	-0.0009	Deflated function value near zero.

Using the same initial estimates each time, you have found four roots for this equation involving a fourth-degree polynomial. However, the last two roots are quite close to each other and are actually one root (with a multiplicity of 2). That is why the root was not eliminated when you tried deflation once at this root. (Round-off error causes the original function to have small positive and negative values for values of x between 8.4999 and 8.5001; for $x = 8.5$ the function is exactly zero.)



Graph of $f(x)$

In general, you will not know in advance the multiplicity of the root you are trying to eliminate. If, after you have attempted to eliminate a root, [SOLVE] finds that same root again, you can proceed in a number of ways:

- Use different initial estimates with the deflated function in an attempt to search for a different root.
- Use deflation again in an attempt to eliminate a multiple root. If you do not know the multiplicity of the root, you may need to repeat this a number of times.

- Examine the behavior of the deflated function at x-values near the known root. If the function's calculated values cross the x-axis smoothly, either another root or a greater multiplicity is indicated.
- Analyze the original function and its derivatives algebraically. It may be possible to determine its behavior for x-values near the known root. (A Taylor series representation, for example, may indicate the multiplicity of a root.)

Limiting the Estimation Time

Occasionally, you may desire to limit the time used by [SOLVE] to find a root. You can use two possible techniques to do this—counting iterations and specifying a tolerance.

Counting Iterations

While searching for a root, [SOLVE] typically samples your function at least a dozen times. Occasionally, [SOLVE] may need to sample it one hundred times or more. (However, [SOLVE] will always stop by itself.) Because your function subroutine is executed once for each estimate that is tried, it can count and limit the number of iterations. An easy way to do this is with an [ISG] instruction to accumulate the number of iterations in the Index register (or other storage register).

If you store an appropriate number in the register before using [SOLVE], your subroutine can interrupt the [SOLVE] algorithm when the limit is exceeded.

Specifying a Tolerance

You can shorten the time required to find a root by specifying a tolerable inaccuracy for your function. Your subroutine should return a function value of zero if the calculated function value is less than the specified tolerance. This tolerance that you specify should correspond to a value that is negligible for practical purposes or should correspond to the accuracy of the computation. This technique eliminates the time required to define the estimate more accurately than is justified by the problem. (The example on page 224 uses this method.)

For Advanced Information

In the *HP-15C Advanced Functions Handbook*, additional, advanced techniques and applications for using [SOLVE] are presented. These topics include:

- Using [SOLVE] with polynomials.
- Solving a system of equations.
- Finding local extremes of a function.
- Using [SOLVE] for financial problems.
- Using [SOLVE] in Complex mode.
- Solving an equation for its complex roots.

Appendix E

A Detailed Look at $\int_y^x$

Section 14, Numerical Integration, presented the basic information you need to use $\int$. This appendix discusses more intricate aspects of $\int$ that are of interest if you use $\int$ often.

How $\int$ Works

The $\int$ algorithm calculates the integral of a function $f(x)$ by computing a weighted average of the function's values at many values of x (known as sample points) within the interval of integration. The accuracy of the result of any such sampling process depends on the number of sample points considered: generally, the more sample points, the greater the accuracy. If $f(x)$ could be evaluated at an infinite number of sample points, the algorithm could—neglecting the limitation imposed by the inaccuracy in the calculated function $f(x)$—provide an exact answer.

Evaluating the function at an infinite number of sample points would take a very long time (namely, forever). However, this is not necessary, since the maximum accuracy of the calculated integral is limited by the accuracy of the calculated function values. Using only a finite number of sample points, the algorithm can calculate an integral that is as accurate as is justified considering the inherent uncertainty in $f(x)$.

The $\int$ algorithm at first considers only a few sample points, yielding relatively inaccurate approximations. If these approximations are not yet as accurate as the accuracy of $f(x)$ would permit, the algorithm is iterated (that is, repeated) with a larger number of sample points. These iterations continue, using about twice as many sample points each time, until the resulting approximation is as accurate as is justified considering the inherent uncertainty in $f(x)$.

The uncertainty of the final approximation is a number derived from the display format, which specifies the uncertainty for the function.* At the end of each iteration, the algorithm compares the approximation calculated during that iteration with the approximations calculated during two previous iterations. If the difference between any of these three approximations and the other two is less than the uncertainty tolerable in the final approximation, the algorithm terminates, placing the current approximation in the X-register and its uncertainty in the Y-register.

It is extremely unlikely that the errors in each of three successive approximations—that is, the differences between the actual integral and the approximations—would all be larger than the disparity among the approximations themselves. Consequently, the error in the final approximation will be less than its uncertainty.† Although we can't know the error in the final approximation, the error is extremely unlikely to exceed the displayed uncertainty of the approximation. In other words, the uncertainty estimate in the Y-register is an almost certain "upper bound" on the difference between the approximation and the actual integral.

Accuracy, Uncertainty, and Calculation Time

The accuracy of an ∫ approximation does not always change when you increase *by just one* the number of digits specified in the display format, though the uncertainty will decrease. Similarly, the time required to calculate an integral sometimes changes when you change the display format, but sometimes does not.

Example: The Bessel function of the first kind, of order four, can be expressed as

$$J_4(x) = \frac{1}{\pi}\int_0^{\pi} \cos(4\theta - x\sin\theta)\, d\theta$$

*The relationship between the display format, the uncertainty in the function, and the uncertainty in the approximation to its integral are discussed later in this appendix.

†Provided that $f(x)$ does not vary rapidly, a consideration that will be discussed in more detail later in this appendix.

Calculate the integral in the expression for $J_4(1)$,

$$\int_0^{\pi} \cos(4\theta - \sin\theta)\, d\theta.$$

First, switch to Program mode and key in a subroutine that evaluates the function $f(\theta) = \cos(4\theta - \sin\theta)$.

Keystrokes	Display	
[g] [P/R]	000-	Program mode.
[f] CLEAR [PRGM]	000-	
[f] [LBL] 0	001-42,21, 0	
4	002- 4	
[×]	003- 20	
[x≷y]	004- 34	
[SIN]	005- 23	
[−]	006- 30	
[COS]	007- 24	
[g] [RTN]	008- 43 32	

Now, switch to Run mode and key the limits of integration into the X- and Y-registers. Be sure the trigonometric mode is set to Radians, and set the display format to [SCI] 2. Finally, press [f] [∫] 0 to calculate the integral.

Keystrokes	Display	
[g] [P/R]		Run mode.
0 [ENTER]	0.0000	Keys lower limit into Y-register.
[g] [π]	3.1416	Keys upper limit into X-register.
[g] [RAD]	3.1416	Sets the trigonometric mode to Radians.
[f] [SCI] 2	3.14 00	Sets display format to [SCI] 2.
[f] [∫] 0	7.79 -03	Integral approximated in [SCI] 2.
[x≷y]	1.45 -03	Uncertainty of [SCI] 2 approximation.

The uncertainty indicates that the displayed digits of the approximation might not include any digits that could be considered accurate. Actually, this approximation is more accurate than its uncertainty indicates.

Keystrokes	Display	
[x⇄y]	7.79 −03	Return approximation to display.
[f] CLEAR [PREFIX] (hold)	7785820888	All 10 digits of [SCI] 2 approximation.

The actual value of this integral, correct to five significant digits, is 7.7805×10^{-3}. Therefore, the error in this approximation is about $(7.7858 - 7.7805) \times 10^{-3} = 5.3 \times 10^{-6}$. This error is considerably less than the uncertainty, 1.45×10^{-3}. The uncertainty is only an *upper bound* on the error in the approximation; the actual error will generally be smaller.

Now calculate the integral in [SCI] 3 and compare the accuracy of the resulting approximation to that of the [SCI] 2 approximation.

Keystrokes	Display	
[f] [SCI] 3	7.786 −03	Changes display format to [SCI] 3.
[R↓] [R↓]	3.142 00	Rolls down stack until upper limit appears in X-register.
[f] [∫] 0	7.786 −03	Integral approximated in [SCI] 3.
[x⇄y]	1.448 −04	Uncertainty of [SCI] 3 approximation.
[x⇄y]	7.786 −03	Returns approximation to display.
[f] CLEAR [PREFIX] (hold)	7785820888	All 10 digits of [SCI] 3 approximation.

All 10 digits of the approximations in [SCI] 2 and [SCI] 3 are identical: the accuracy of the approximation in [SCI] 3 is no better than the accuracy in [SCI] 2 despite the fact that the uncertainty in [SCI] 3 is less than the uncertainty in [SCI] 2. Why is this? Remember that the accuracy of any approximation depends primarily on the number of sample points at which the function $f(x)$ has been evaluated. The [∫] algorithm is iterated with increasing numbers of sample points until the disparity among three successive approximations is less than the uncertainty derived from the display format. After a particular iteration, the disparity among the approximations may already be so much less than the uncertainty that it would still be less if the uncertainty were decreased by a factor of 10. In such cases, if you decreased the uncertainty by specifying one more digit in the display format, the algorithm would not have to consider additional sample points, and the resulting approximation would be identical to the approximation calculated with the larger uncertainty.

If you calculated the two preceding approximations on your calculator, you may have noticed that it did not take any longer to calculate the integral in [SCI] 3 than in [SCI] 2. This is because the time to calculate the integral of a given function depends on the number of sample points at which the function must be evaluated to achieve an approximation of acceptable accuracy. For the [SCI] 3 approximation, the algorithm did not have to consider more sample points than it did in [SCI] 2, so it did not take any longer to calculate the integral.

Often, however, increasing the number of digits in the display format will require evaluating the function at additional sample points, so that calculating the integral will take more time. Now calculate the same integral in [SCI] 4.

Keystrokes	Display		
[f][SCI]4	**7.7858**	**-03**	[SCI] 4 display.
[R↓][R↓]	**3.1416**	**00**	Rolls down stack until upper limit appears in X-register.
[f][∫]0	**7.7807**	**-03**	Integral approximated in [SCI] 4.

This approximation took about twice as long as the approximation in SCI 3 or SCI 2. In this case, the algorithm had to evaluate the function at about twice as many sample points as before in order to achieve an approximation of acceptable accuracy. Note, however, that you received a reward for your patience: the accuracy of this approximation is better, by almost two digits, than the accuracy of the approximation calculated using half the number of sample points.

The preceding examples show that repeating the approximation of an integral in a different display format sometimes will give you a more accurate answer, but sometimes it will not. Whether or not the accuracy is changed depends on the particular function, and generally can be determined only by trying it.

Furthermore, if you do get a more accurate answer, it will come at the cost of about double the calculation time. This unavoidable trade-off between accuracy and calculation time is important to keep in mind if you are considering decreasing the uncertainty in hopes of obtaining a more accurate answer.

The time required to calculate the integral of a given function depends not only on the number of digits specified in the display format, but also, to a certain extent on the limits of integration. When the calculation of an integral requires an excessive amount of time, the width of the interval of integration (that is, the difference of the limits) may be too large compared with certain features of the function being integrated. For most problems, however, you need not be concerned about the effects of the limits of integration on the calculation time. These conditions, as well as techniques for dealing with such situations, will be discussed later in this appendix.

Uncertainty and the Display Format

Because of round-off error, the subroutine you write for evaluating $f(x)$ cannot calculate $f(x)$ exactly, but rather calculates

$$\hat{f}(x) = f(x) \pm \delta_1(x),$$

where $\delta_1(x)$ is the uncertainty of $f(x)$ caused by round-off error. If

$f(x)$ relates to a physical situation, then the function you would like to integrate is not $f(x)$ but rather

$$F(x) = f(x) \pm \delta_2(x),$$

where $\delta_2(x)$ is the uncertainty associated with $f(x)$ that is caused by the approximation to the actual physical situation.

Since $f(x) = \hat{f}(x) \pm \delta_1(x)$, the function you want to integrate is

$$F(x) = \hat{f}(x) \pm \delta_1(x) \pm \delta_2(x)$$

or

$$F(x) = \hat{f}(x) \pm \delta(x),$$

where $\delta(x)$ is the net uncertainty associated with $\hat{f}(x)$.

Therefore, the integral you want is

$$\int_a^b F(x)\,dx = \int_a^b [\hat{f}(x) \pm \delta(x)]\,dx$$

$$= \int_a^b \hat{f}(x)\,dx \pm \int_a^b \delta(x)\,dx$$

$$= I \pm \Delta$$

where I is the approximation to $\int_a^b F(x)\,dx$ and Δ is the uncertainty associated with the approximation. The [∫] algorithm places the number I in the X-register and the number Δ in the Y-register.

The uncertainty $\delta(x)$ of $\hat{f}(x)$, the function calculated by your subroutine, is determined as follows. Suppose you consider three significant digits of the function's values to be accurate, so you set the display format to [SCI] 2. The display would then show only the accurate digits in the mantissa of a function's values: for example, **1.23 -04.**

Since the display format rounds the number in the X-register to the number displayed, this implies that the uncertainty in the function's values is $\pm 0.005 \times 10^{-4} = \pm 0.5 \times 10^{-2} \times 10^{-4} =$

$\pm 0.5 \times 10^{-6}$. Thus, setting the display format to SCI n or ENG n, where n is an integer,* implies that the uncertainty in the function's values is

$$\delta(x) = 0.5 \times 10^{-n} \times 10^{m(x)}$$

$$= 0.5 \times 10^{-n+m(x)}$$

In this formula, n is the number of digits specified in the display format and $m(x)$ is the exponent of the function's value at x that would appear if the value were displayed in SCI display format.

The uncertainty is proportional to the factor $10^{m(x)}$, which represents the magnitude of the function's value at x. Therefore, SCI and ENG display formats imply an uncertainty in the function that is *relative* to the function's magnitude.

Similarly, if a function value is displayed in FIX n, the rounding of the display implies that the uncertainty in the function's values is

$$\delta(x) = 0.5 \times 10^{-n}.$$

Since this uncertainty is independent of the function's magnitude, FIX display format implies an uncertainty that is *absolute*.

Each time the ∫ algorithm samples the function at a value of x, it also derives a sample of $\delta(x)$, the uncertainty of the function's value at x. This is calculated using the number of digits n currently specified in the display format and (if the display format is set to

---

* Although SCI 8 or 9 generally results in the same *display* as SCI 7, it will result in a smaller *uncertainty* of a calculated integral. (The same is true for the ENG format.) A negative value for n (which can be set by using the Index register) will also affect the uncertainty of an ∫ calculation. The minimum value for n that will affect uncertainty is –6. A number in R_I less than –6 will be interpreted as –6.

SCI or ENG) the magnitude $m(x)$ of the function's value at x. The number Δ, the uncertainty of the approximation to the desired integral, is the integral of $\delta(x)$:

$$\Delta = \int_a^b \delta(x)\,dx$$

$$= \int_a^b [0.5 \times 10^{-n+m(x)}]\,dx.$$

This integral is calculated using the samples of $\delta(x)$ in roughly the same ways that the approximation to the integral of the function is calculated using the samples of $\hat{f}(x)$.

Because Δ is proportional to the factor 10^{-n}, the uncertainty of an approximation changes by about a factor of 10 for each digit specified in the display format. This will generally not be exact in SCI or ENG display format, however, because changing the number of digits specified may require that the function be evaluated at different sample points, so that $\delta(x) \sim 10^{m(x)}$ would have different values.

Note that when an integral is approximated in FIX display format, $m(x) = 0$ and so the calculated uncertainty in the approximation turns out to be

$$\Delta = 0.5 \times 10^{-n}(b - a).$$

Normally you do not have to determine precisely the uncertainty in the function. (To do so would frequently require a very complicated analysis.) Generally, it's more convenient to use SCI or ENG display format if the uncertainty in the function's values can be more easily estimated as a *relative* uncertainty. On the other hand, it's more convenient to use FIX display format if the uncertainty in the function's values can be more easily estimated as an *absolute uncertainty*. FIX display format may be inappropriate to use (leading to peculiar results) when you are integrating a function whose magnitude *and* uncertainty have extremely small values within the interval of integration. Likewise, SCI display format may be inappropriate to use (also leading to peculiar results) if the magnitude of the function becomes much smaller than its uncertainty. If the results of calculating an integral seem strange,

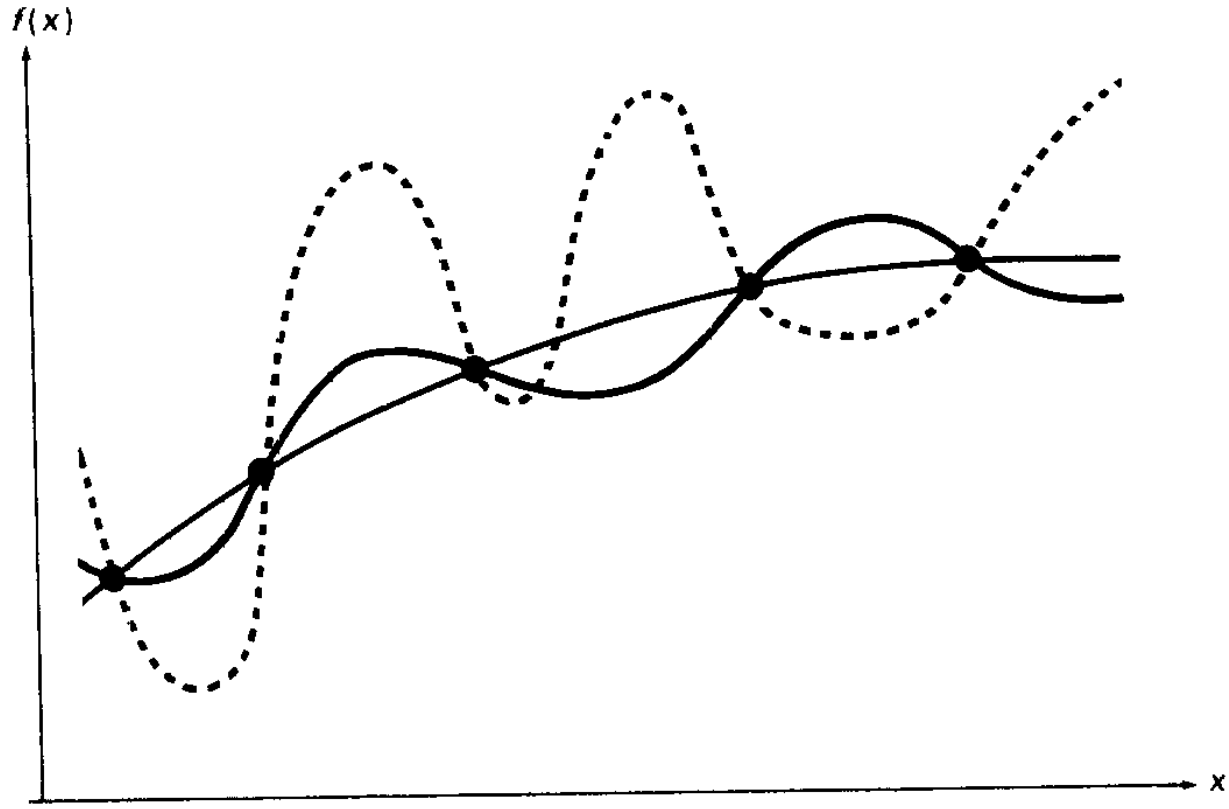
it may be more appropriate to calculate the integral in the alternate display format.

Conditions That Could Cause Incorrect Results

Although the ∫ algorithm in the HP-15C is one of the best available, in certain situations it—like nearly all algorithms for numerical integration—might give you an incorrect answer. *The possibility of this occurring is extremely remote*. The ∫ algorithm has been designed to give accurate results with almost any smooth function. Only for functions that exhibit *extremely* erratic behavior is there any substantial risk of obtaining an inaccurate answer. Such functions rarely occur in problems related to actual physical situations; when they do, they usually can be recognized and dealt with in a straightforward manner.

As discussed on page 240, the ∫ algorithm samples the function $f(x)$ at various values of x within the interval of integration. By calculating a weighted average of the function's values at the sample points, the algorithm approximates the integral of $f(x)$.

Unfortunately, since all that the algorithm knows about $f(x)$ are its values at the sample points, it cannot distinguish between $f(x)$ and any other function that agrees with $f(x)$ at all the sample points. This situation is depicted in the illustration on the next page, which shows (over a portion of the interval of integration) three of the infinitely many functions whose graphs include the finitely many sample points.



With this number of sample points, the algorithm will calculate the same approximation for the integral of any of the functions shown. The actual integrals of the functions shown with solid lines are about the same, so the approximation will be fairly accurate if $f(x)$ is one of these functions. However, the actual integral of the function shown with a dashed line is quite different from those of the others, so the current approximation will be rather inaccurate if $f(x)$ is this function.

The [∫] algorithm comes to know the general behavior of the function by sampling the function at more and more points. If a fluctuation of the function in one region is not unlike the behavior over the rest of the interval of integration, at some iteration the algorithm will likely detect the fluctuation. When this happens, the number of sample points is increased until successive iterations yield approximations that take into account the presence of the most rapid, *but characteristic*, fluctuations.

For example, consider the approximation of

$$\int_0^\infty xe^{-x}dx.$$

Since you're evaluating this integral numerically, you might think (naively in this case, as you'll see) that you should represent the upper limit of integration by 10^{99}—which is virtually the largest number you can key into the calculator. Try it and see what happens.

Key in a subroutine that evaluates the function $f(x) = xe^{-x}$.

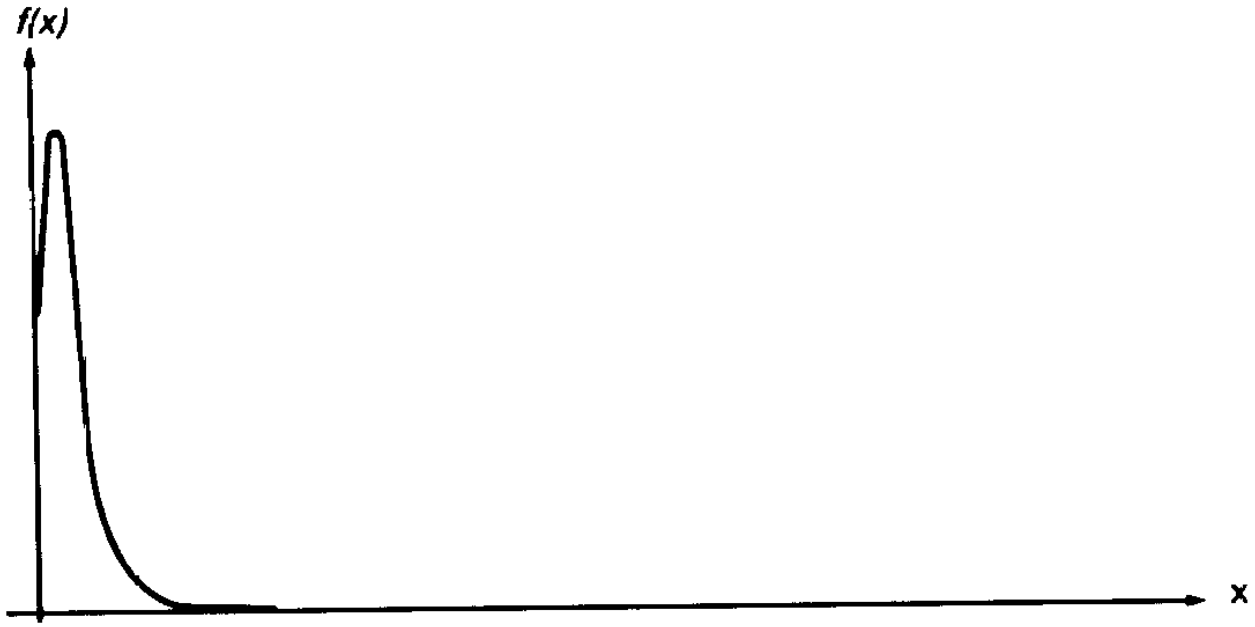
Keystrokes	Display	
[g] [P/R]	000-	Program mode.
[f] [LBL] 1	001-42,21, 1	
[CHS]	002- 16	
[e^x]	003- 12	
[×]	004- 20	
[g] [RTN]	005- 43 32	

Set the calculator to Run mode. Then set the display format to [SCI] 3 and key the limits of integration into the X- and Y-registers.

Keystrokes	Display	
[g] [P/R]		Run mode.
[f] [SCI] 3		Sets display format to [SCI] 3.
0 [ENTER]	0.000 00	Keys lower limit into Y-register.
[EEX] 99	1 99	Keys upper limit into X-register.
[f] [∫] 1	0.000 00	Approximation of integral.

The answer returned by the calculator is clearly incorrect, since the actual integral of $f(x) = xe^{-x}$ from 0 to ∞ is exactly 1. But the problem is *not* that you represented ∞ by 10^{99}, since the actual

integral of this function from 0 to 10^{99} is very close to 1. The reason you got an incorrect answer becomes apparent if you look at the graph of $f(x)$ over the interval of integration:

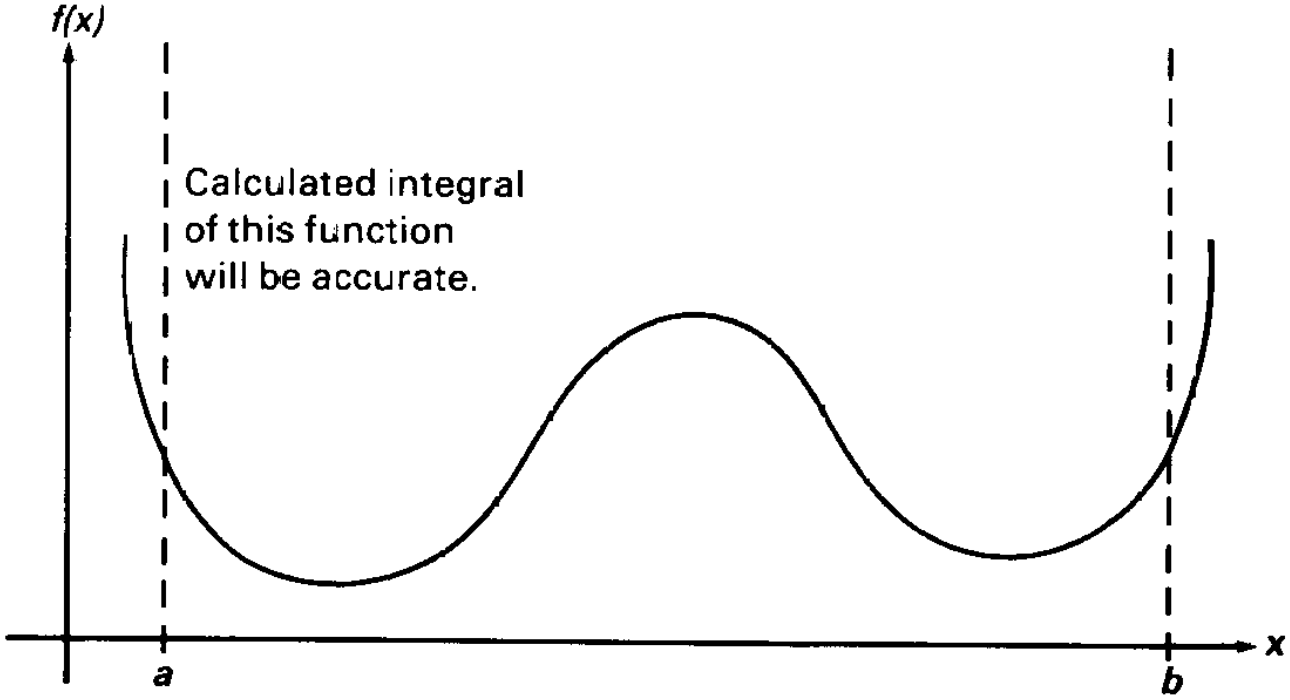


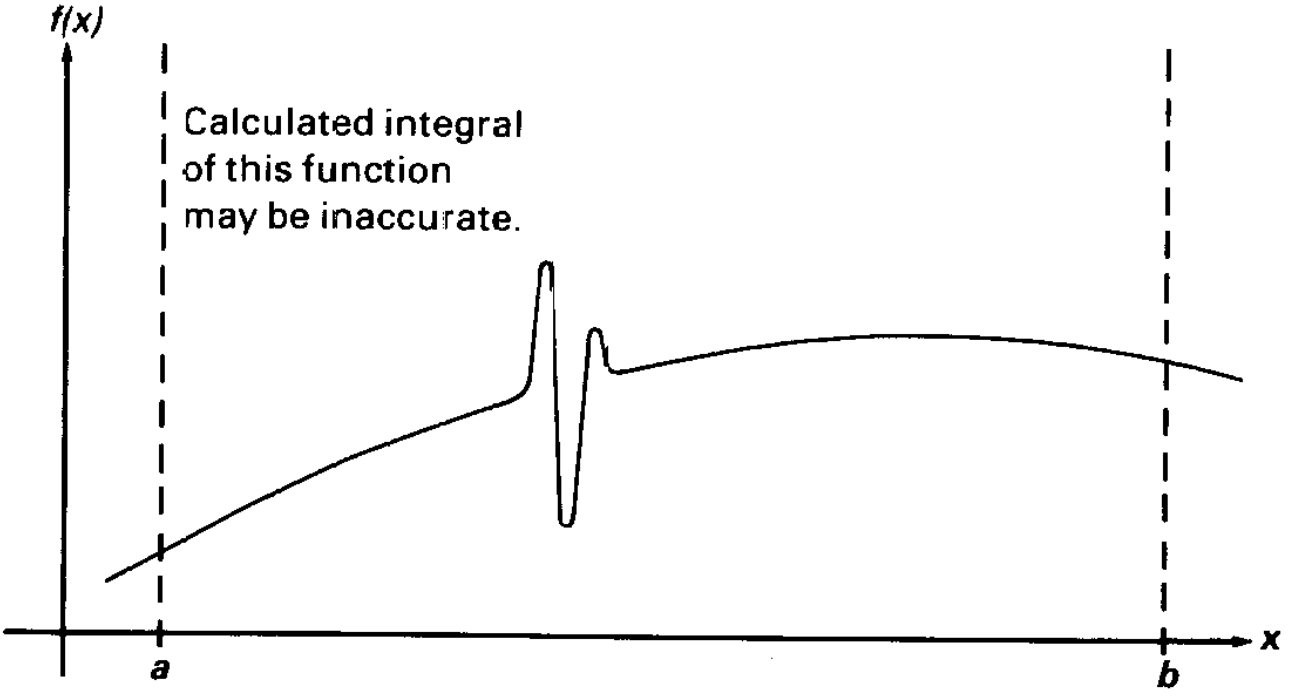
The graph is a spike very close to the origin. (Actually, to illustrate $f(x)$ the width of the spike has been considerably exaggerated. Shown in actual scale over the interval of integration, the spike would be indistinguishable from the vertical axis of the graph.) Because no sample point happened to discover the spike, the algorithm assumed that $f(x)$ was identically equal to zero throughout the interval of integration. Even if you increased the number of sample points by calculating the integral in [SCI] 9, none of the additional sample points would discover the spike when this particular function is integrated over this particular interval. (Better approaches to problems such as this are mentioned at the end of the next topic, Conditions That Prolong Calculation Time.)

You've seen how the [f] algorithm can give you an incorrect answer when $f(x)$ has a fluctuation somewhere that is very uncharacteristic of the behavior of the function elsewhere. Fortunately, functions exhibiting such aberrations are unusual enough that you are unlikely to have to integrate one unknowingly.

Functions that could lead to incorrect results can be identified in simple terms by how rapidly it and its low-order derivatives vary across the interval of integration. Basically, the more rapid the variation in the function or its derivatives, and the lower the order of such rapidly varying derivatives, the less quickly will the [f] algorithm terminate, and the less reliable will the resulting approximation be.

Note that the rapidity of variation in the function (or its low-order derivatives) must be determined with respect to the width of the interval of integration. With a given number of sample points, a function $f(x)$ that has three fluctuations can be better characterized by its samples when these variations are spread out over most of the interval of integration than if they are confined to only a small fraction of the interval. (These two situations are shown in the next two illustrations.) Considering the variations or fluctuations as a type of oscillation in the function, the criterion of interest is the ratio of the period of the oscillations to the width of the interval of integration: the larger this ratio, the more quickly the algorithm will terminate, and the more reliable will be the resulting approximation.





In many cases you will be familiar enough with the function you want to integrate that you'll know whether the function has any quick wiggles relative to the interval of integration. If you're not familiar with the function, and you have reason to suspect that it may cause problems, you can quickly plot a few points by evaluating the function using the subroutine you wrote for that purpose.

If for any reason, after obtaining an approximation to an integral, you have reason to suspect its validity, there's a very simple procedure you can use to verify it: subdivide the interval of integration into two or more adjacent subintervals, integrate the function over each subinterval, then add the resulting approximations. This causes the function to be sampled at a brand new set of sample points, thereby more likely revealing any previously hidden spikes. If the initial approximation was valid, it will equal the sum of the approximations over the subintervals.

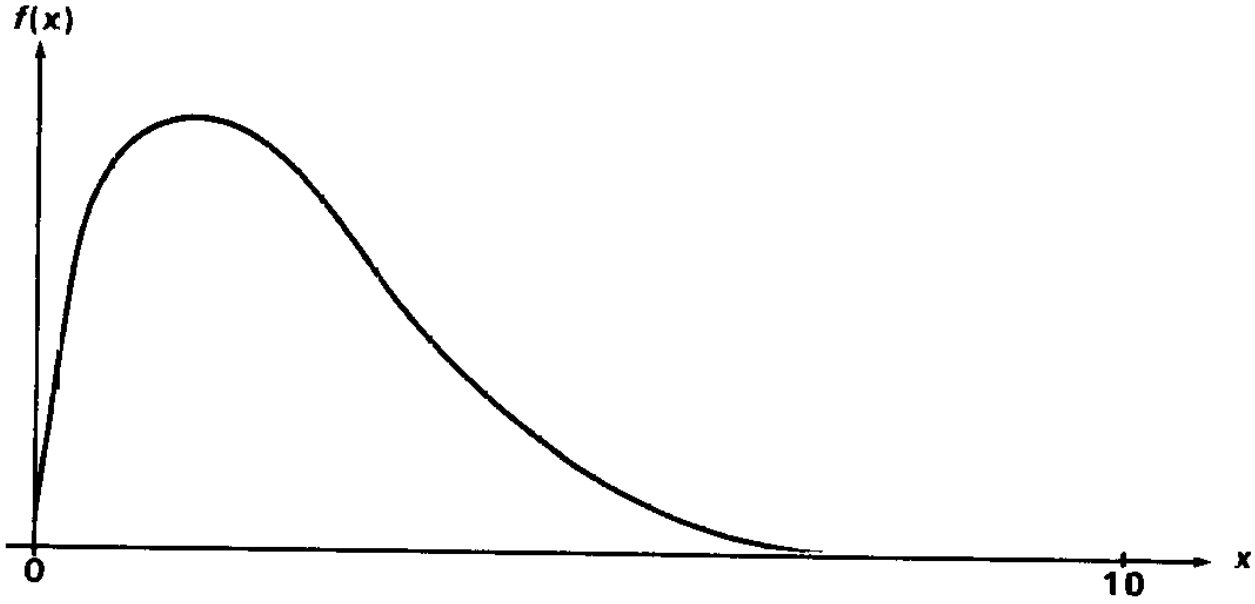
Conditions That Prolong Calculation Time

In the preceding example (page 251), you saw that the algorithm gave an incorrect answer because it never detected the spike in the function. This happened because the variation in the function was too quick relative to the width of the interval of integration. If the width of the interval were smaller, you would get the correct answer; but it would take a very long time if the interval were still too wide.

For certain integrals such as the one in that example, calculating the integral may be unduly prolonged because the width of the interval of integration is too large relative to certain features of the functions being integrated. Consider an integral where the interval of integration is wide enough to require excessive calculation time but not so wide that it would be calculated incorrectly. Note that because $f(x) = xe^{-x}$ approaches zero very quickly as x approaches ∞, the contribution to the integral of the function at large values of x is negligible. Therefore, you can evaluate the integral by replacing ∞, the upper limit of integration, by a number not so large as 10^{99}, say 10^3.

Keystrokes	Display	
0 ENTER	0.000 00	Keys lower limit into Y-register.
EEX 3	1 03	Keys upper limit into X-register.
f ∫ 1	1.000 00	Approximation to integral.
x≷y	1.824 -04	Uncertainty of approximation.

This is the correct answer, but it took a very long time. To understand why, compare the graph of the function over the interval of integration, which looks about identical to that shown on page 252, to the graph of the function between $x = 0$ and $x = 10$.



By comparing the two graphs, you can see that the function is "interesting" only at small values of x. At greater values of x, the function is "uninteresting," since it decreases smoothly and gradually in a very predictable manner.

As discussed earlier, the ∫ algorithm will sample the function with higher densities of sample points until the disparity between successive approximations becomes sufficiently small. In other words, the algorithm samples the function at increasing numbers of sample points until it has sufficient information about the function to provide an approximation that changes insignificantly when further samples are considered.

If the interval of integration were (0, 10) so that the algorithm needed to sample the function only at values where it was interesting but relatively smooth, the sample points after the first few iterations would contribute no new information about the behavior of the function. Therefore, only a few iterations would be necessary before the disparity between successive approximations became sufficiently small that the algorithm could terminate with an approximation of a given accuracy.

On the other hand, if the interval of integration were more like the one shown in the graph on page 252, most of the sample points would capture the function in the region where its slope is not varying much. The few sample points at small values of x would find that values of the function changed appreciably from one iteration to the next. Consequently the function would have to be evaluated at additional sample points before the disparity between successive approximations would become sufficiently small.

In order for the integral to be approximated with the same accuracy over the larger interval as over the smaller interval, the density of the sample points must be the same in the region where the function is interesting. To achieve the same density of sample points, the total number of sample points required over the larger interval is much greater than the number required over the smaller interval. Consequently, several more iterations are required over the larger interval to achieve an approximation with the same accuracy, and therefore calculating the integral requires considerably more time.

Because the calculation time depends on how soon a certain density of sample points is achieved in the region where the function is interesting, the calculation of the integral of any function will be prolonged if the interval of integration includes mostly regions where the function is not interesting. Fortunately, if you must calculate such an integral, you can modify the problem so that the calculation time is considerably reduced. Two such techniques are subdividing the interval of integration and transformation of variables. These methods enable you to change the function or the limits of integration so that the integrand is better behaved over the interval(s) of integration. (These techniques are described in the *HP-15C Advanced Functions Handbook*.)

Obtaining the Current Approximation to an Integral

When the calculation of an integral is requiring more time than you care to wait, you may want to stop and display the current approximation. You can obtain the current approximation, but not its uncertainty.

Pressing R/S while the HP-15C is calculating an integral halts the calculation, just as it halts the execution of a running program. When you do so, the calculator stops at the current program line in the subroutine you wrote for evaluating the function, and displays the result of executing the preceding program line. Note that after you halt the calculation, the current approximation to the integral is *not* the number in the X-register nor the number in any other stack register. Just as with any program, pressing R/S again starts the calculation from the program line at which it was stopped.

The ∫f algorithm updates the current approximation and stores it in the LAST X register after evaluating the function at each new sample point. To obtain the current approximation, therefore, simply halt the calculator, single-step if necessary through your function subroutine until the calculator has finished evaluating the function and updating the current approximation. Then recall the contents of the LAST X register, which are updated when the RTN instruction in the function subroutine is executed.

While the calculator is updating the current approximation, the display is blank and does not show **running**. (While the calculator is executing your function subroutine, **running** is displayed.) Therefore, you might avoid having to single-step through your subroutine by halting the calculator at a moment when the display is blank.

In summary, to obtain the current approximation to an integral, follow the steps below.

1. Press R/S to halt the calculator, preferably while the display is blank.
2. When the calculator halts, switch to Program mode to check the current program line.
   - If that line contains the subroutine label, return to Run mode and view the LAST X register (step 3).

- If any other program line is displayed, return to Run mode and single-step (SST) through the program until you reach a RTN instruction (keycode 43 32) or line 000 (if there is no RTN). (Be sure to hold the SST key down long enough to view the program line numbers and keycodes.)

3. Press g LSTx to view the current approximation. If you want to continue calculating the final approximation, press ← + R/S. This refills the stack with the current x-value and restarts the calculator.

For Advanced Information

The *HP-15C Advanced Functions Handbook* explores more esoteric aspects of ∫ and its applications. These topics include:

- Accuracy of the function to be integrated.
- Shortening calculation time.
- Calculating difficult integrals.
- Using ∫ in Complex mode.

Appendix F

Battery, Warranty, and Service Information

Batteries

The HP-15C is powered by three batteries. In "typical" use, the HP-15C has been designed to operate 6 months or more on a set of alkaline batteries. The batteries supplied with the calculator are alkaline, but silver-oxide batteries (which should last twice as long) can also be used.

A set of three fresh alkaline batteries will provide at least 60 hours of *continuous* program running (the most power-consuming kind of calculator use*). A set of three fresh silver-oxide batteries will provide at least 135 hours of *continuous* program running. If the calculator is being used to perform operations other than running programs, it uses much less power. When only the display is on—that is, if you are not pressing keys or running programs—very little power is consumed.

If the calculator remains turned off, a set of fresh batteries will preserve the contents of Continuous Memory for as long as the batteries would last outside of the calculator—at least 1½ years for alkaline batteries or at least 2 years for silver-oxide batteries.

The actual lifetime of the batteries depends on how often you use the calculator, whether you use it more for running programs or more for manual calculations, and which functions you use.*

The batteries supplied with the calculator, as well as the batteries listed on the next page for replacement, are *not* rechargeable.

---

* Power consumption in the HP-15C depends on the mode of calculator use: off (with Continuous Memory preserved); idle (with only the display on); or "operating" (running a program, performing a calculation, or having a key pressed). While the calculator is turned on, typical calculator use is a mixture of idle time and "operating" time. Therefore, the actual lifetime of the batteries depends on how much time the calculator spends in each of the three modes.

WARNING

Do not attempt to recharge the batteries; do not store batteries near a source of high heat; do not dispose of batteries in fire. Doing so may cause the batteries to leak or explode.

The following batteries are recommended for replacement in your HP-15C (not all batteries available in all countries):

Alkaline	Silver-Oxide
Eveready A76*	Eveready 357*
UCAR A76	UCAR 357
National or Panasonic LR44	

Low-Power Indication

An asterisk (*) flashing in the lower left corner of the display when the calculator is on signifies that the available battery power is running low.

With alkaline batteries installed:

- The calculator can be used for at least 1½ hours of continuous program running after the asterisk first appears.†
- If the calculator remains turned off, the contents of its Continuous Memory will be preserved for at least 1 month after the asterisk first appears.

With silver-oxide batteries installed:

- The calculator can be used for at least 10 minutes of continuous program running after the asterisk first appears.†
- If the calculator remains turned off, the contents of its

---

* Not available in the United Kingdom or Republic of Ireland.

† Note that this time is the minimum available for *continuous program running*—that is, while continuously "operating" (as described in the footnote on page 259). If you are using the calculator for manual calculations—a mixture of the idle and "operating" modes—the calculator can be used for a much longer time after the asterisk first appears.

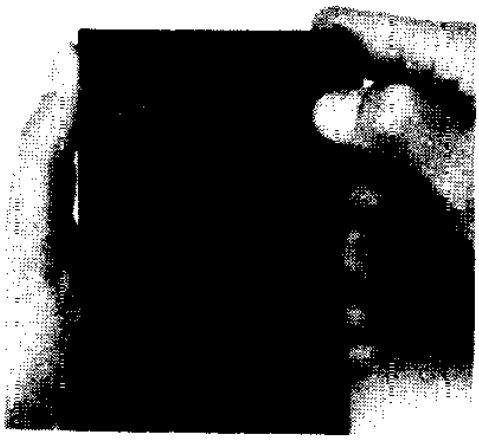
Continuous Memory will be preserved for at least 1 week after the asterisk first appears.

Installing New Batteries

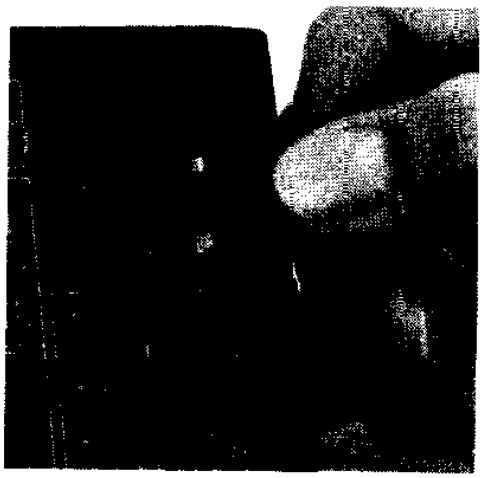
The contents of the calculator's Continuous Memory are preserved for a short time while the batteries are out of the calculator (provided that you turn off the calculator before removing the batteries). This allows you ample time to replace the batteries without losing data or programs. If the batteries are left out of the calculator for an extended period, the contents of Continuous Memory may be lost.

To install new batteries, use the following procedure:

1. Be sure that the calculator is off.
2. Holding the calculator as shown, press outward on the battery compartment door until it opens slightly.



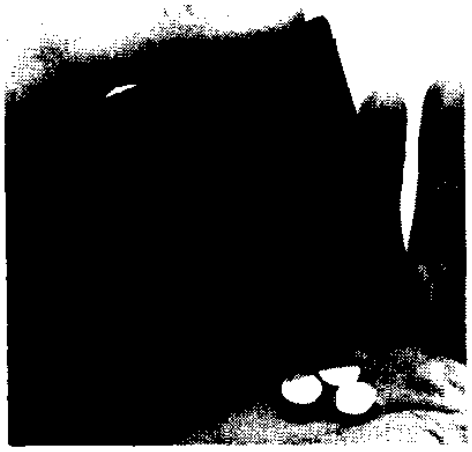
3. Grasp the outer edge of the battery compartment door, then tilt it up and out of the calculator.



CAUTION

In the next two steps, be careful not to press any keys while batteries are out of the calculator. If you do so, the contents of Continuous Memory may be lost and keyboard control may be lost (that is, the calculator may not respond to keystrokes).

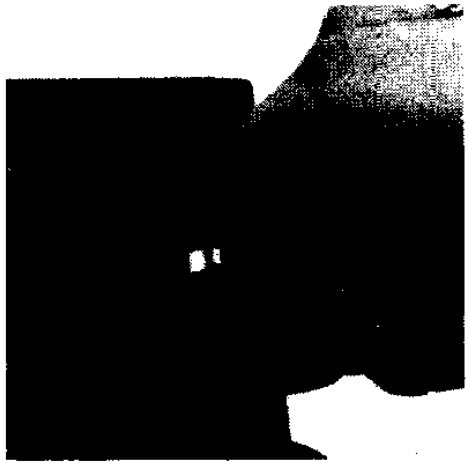
4. Turn the calculator over and gently shake, allowing the batteries to fall into the palm of your hand.



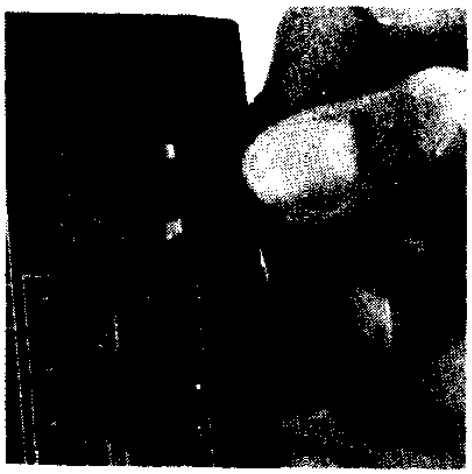
CAUTION

In the next step, replace *all three* batteries with fresh ones. If you leave an old battery inside, it may leak. Furthermore, be careful not to insert the batteries backwards. If you do so, the contents of Continuous Memory may be lost, and the batteries may be damaged.

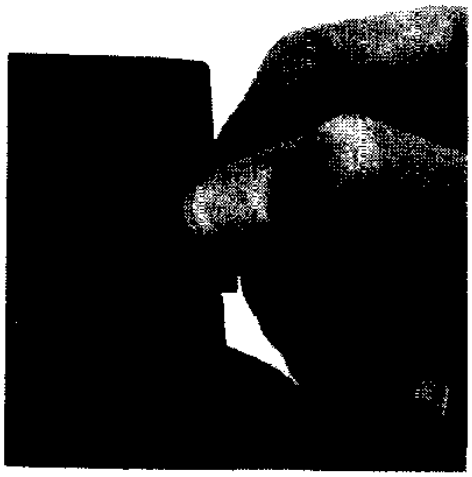
5. Insert three new batteries under the plastic flap or flaps shielding the battery compartment. They should be positioned with their flat sides (the sides marked +) facing *toward* the nearby rubber foot, as shown in the illustration on the calculator case.



6. Insert the tab of the battery compartment door into the slot in the calculator case.



7. Lower the battery compartment door until it is flush with the case, then push the door inward until it is tightly shut.
8. Turn the calculator on. If for any reason Continuous Memory has been reset (that is, if its contents have been lost), the display will show **Pr Error**. Pressing any key will clear this message from the display.



Verifying Proper Operation (Self-Tests)

If it appears that the calculator will not turn on or otherwise is not operating properly, use one of the following procedures.

For a calculator that does *not* respond to keystrokes:

1. Press the [y^x] and [ON] keys simultaneously, then release them. This will alter the contents of the X-register, so clear the X-register afterward.
2. If the calculator still does not respond to keystrokes, remove and reinsert the batteries. Make sure the batteries are properly positioned in the compartment.
3. If the calculator still does not respond to keystrokes, *leave the batteries in the compartment* and short the battery terminals together. (The batteries must remain in place to prevent possible internal damage to the calculator.) With a paper clip or piece of wire, briefly connect the terminals.

Only momentary contact is required. The terminals are matching metal strips, or a combination of one spring and one hard edged tab located at either end of the battery compartment. After you do this, the contents of Continuous Memory will be lost, and you may need to press the [ON] key more than once to turn the calculator back on.

4. If the calculator still does not turn on, install fresh batteries. If there is still no response, the calculator requires service.

For a calculator that *does* respond to keystrokes:

1. With the calculator off, hold down the [ON] key and press [×].
2. Release the [ON] key, then release the [×] key. This initiates a complete test of the calculator's electronic circuitry. If everything is working correctly, within about 25 seconds (during which the word **running** flashes) the display should show **-8.8.8.8.8.8.8.8.8.8.**, and all of the status indicators (except the * low-power indicator) should turn on.* If the display shows **Error 9**, goes blank, or otherwise does not show the proper result, the calculator requires service.†

Note: Tests of the calculator's electronics are also performed if the [+] key or the [÷] key is held down when [ON] is released. †‡ These tests are included in the calculator to be used in verifying that it is operating properly during manufacture and service.

* The status indicators turned on at the end of this test include some that normally are not displayed on the HP-15C.

† If the calculator displays **Error 9** as a result of the [ON]/[×] test or the [ON]/[+] test but you wish to continue using your calculator, you should reset Continuous Memory as described on page 63.

‡ The [ON]/[+] combination initiates a test that is similar to that described above, but continues indefinitely. The test can be terminated by pressing any key, which will halt the test within 25 seconds. The [ON]/[÷] combination initiates a test of the keyboard and the display. When the [ON] key is released, certain segments in the display will be lit. To run the test, the keys are pressed in order from left to right along each row, from the top row to the bottom row. As each key is pressed, different segments in the display are lit. If the calculator is operating properly *and all the keys are pressed in the proper order*, the calculator will display **15** after the last key is pressed. (The [ENTER] key should be pressed both with the third-row keys and with the fourth-row keys.) If the calculator is not working properly, *or if a key is pressed out of order*, the calculator will display **Error 9**. *Note that if this error display results from an incorrect key being pressed, this does not indicate that your calculator requires service.* This test can be terminated by pressing any key out of order (which will, of course, result in the **Error 9** display). Both the **Error 9** display and the **15** display can be cleared by pressing any key.

If you had suspected that the calculator was not working properly but the proper display was obtained in step 2, it is likely that you made an error in operating the calculator.We suggest you reread the section in this handbook applicable to your calculation. If you still experience difficulty, write or telephone Hewlett-Packard at an address or phone number listed under Service (page 267).

Limited One-Year Warranty

What We Will Do

The HP-15C (except for the batteries, or damage caused by the batteries) is warranted by Hewlett-Packard against defects in materials and workmanship for one year from the date of original purchase. If you sell your unit or give it as a gift, the warranty is automatically transferred to the new owner and remains in effect for the original one-year period. During the warranty period, we will repair or, at our option, replace at no charge a product that proves to be defective, provided you return the product, shipping prepaid, to a Hewlett-Packard service center.

What Is Not Covered

Batteries, and damage caused by the batteries, are not covered by the Hewlett-Packard warranty. Check with the battery manufacturer about battery and battery leakage warranties.

This warranty does not apply if the product has been damaged by accident or misuse or as the result of service or modification by other than an authorized Hewlett-Packard service center.

No other express warranty is given. The repair or replacement of a product is your exclusive remedy. **ANY OTHER IMPLIED WARRANTY OF MERCHANTABILITY OR FITNESS IS LIMITED TO THE ONE-YEAR DURATION OF THIS WRITTEN WARRANTY.** Some states, provinces, or countries do not allow limitations on how long an implied warranty lasts, so the above limitation may not apply to you. **IN NO EVENT SHALL HEWLETT-PACKARD COMPANY BE LIABLE FOR CONSEQUENTIAL DAMAGES.** Some states, provinces, or countries do not allow the exclusion or limitation of incidental or consequential damages, so the above limitation or exclusion may not apply to you.

This warranty gives you specific legal rights, and you may also have other rights which vary from state to state, province to province, or country to country.

Warranty for Consumer Transactions in the United Kingdom

This warranty shall not apply to consumer transactions and shall not affect the statutory rights of a consumer. In relation to such transactions, the rights and obligations of Seller and Buyer shall be determined by statute.

Obligation to Make Changes

Products are sold on the basis of specifications applicable at the time of manufacture. Hewlett-Packard shall have no obligation to modify or update products once sold.

Warranty Information

If you have any questions concerning this warranty, please contact:

- In the United States:

Hewlett-Packard
Calculator Service Center
1030 N.E. Circle Blvd.
Corvallis, OR 97330
Telephone: (503) 757-2002

- In Europe:

Hewlett-Packard S.A.
150, route du Nant-d'Avril
P.O. Box
CH-1217 Meyrin 2
Geneva
Switzerland
Telephone: (022) 83 81 11

Note: Do *not* send calculators to this address for repair.

- In other countries:

Hewlett-Packard Intercontinental
3495 Deer Creek Rd.
Palo Alto, California 94304
U.S.A.
Telephone: (415) 857-1501

Note: Do *not* send calculators to this address for repair.

Service

Hewlett-Packard maintains service centers in most major countries throughout the world. You may have your unit repaired at a Hewlett-Packard service center any time it needs service, whether the unit is under warranty or not. There is a charge for repairs after the one-year warranty period.

Hewlett-Packard calculator products normally are repaired and reshipped within five (5) working days of receipt at any service center. This is an average time and could vary depending upon the time of year and work load at the service center. The total time you are without your unit will depend largely on the shipping time.

Obtaining Repair Service in the United States

The Hewlett-Packard United States Service Center for handheld and portable calculator products is located in Corvallis, Oregon:

Hewlett-Packard Company
Service Department
P.O. Box 999 *or* 1030 N.E. Circle Blvd.
Corvallis, OR 97330, U.S.A.
Telephone: (503) 757-2000

Obtaining Repair Service in Europe

Service centers are maintained at the following locations. For countries not listed, contact the dealer where you purchased your calculator.

AUSTRIA
HEWLETT-PACKARD Ges.m.b.H.
Kleinrechner-Service
Wagramerstrasse-Lieblgasse 1
A-1220 Wien (Vienna)
Telephone: (0222) 23 65 11

BELGIUM
HEWLETT-PACKARD BELGIUM SA/NV
Woluwedal 100
B-1200 Brussels
Telephone: (02) 762 32 00

DENMARK
HEWLETT-PACKARD A/S
Datavej 52
DK-3460 Birkerod (Copenhagen)
Telephone: (02) 81 66 40

EASTERN EUROPE
Refer to the address listed under Austria.

FINLAND
HEWLETT-PACKARD OY
Revontulentie 7
SF-02100 Espoo 10 (Helsinki)
Telephone: (90) 455 02 11

FRANCE
HEWLETT-PACKARD FRANCE
Division Informatique Personnelle
S.A.V. Calculateurs de Poche
F-91947 Les Ulis Cedex
Telephone: (6) 907 78 25

GERMANY
HEWLETT-PACKARD GmbH
Kleinrechner-Service
Vertriebszentrale
Berner Strasse 117
Postfach 560 140
D-6000 Frankfurt 56
Telephone: (611) 50041

ITALY
HEWLETT-PACKARD ITALIANA S.P.A.
Casella postale 3645 (Milano)
Via G. Di Vittorio, 9
I-20063 Cernusco Sul Naviglio (Milan)
Telephone: (2) 90 36 91

NETHERLANDS
HEWLETT-PACKARD NEDERLAND B.V.
Van Heuven Goedhartlaan 121
NL-1181 KK Amstelveen (Amsterdam)
P.O. Box 667
Telephone: (020) 472021

NORWAY
HEWLETT-PACKARD NORGE A/S
P.O. Box 34
Oesterndalen 18
N-1345 Oesteraas (Oslo)
Telephone: (2) 17 11 80

SPAIN
HEWLETT-PACKARD ESPANOLA S.A.
Calle Jerez 3
E-Madrid 16
Telephone: (1) 458 2600

SWEDEN
HEWLETT-PACKARD SVERIGE AB
Skalholtsgatan 9, Kista
Box 19
S-163 93 Spanga (Stockholm)
Telephone: (08) 750 20 00

SWITZERLAND
HEWLETT-PACKARD (SCHWEIZ) AG
Kleinrechner-Service
Allmend 2
CH-8967 Widen
Telephone: (057) 31 21 11

UNITED KINGDOM
HEWLETT-PACKARD Ltd
King Street Lane
GB-Winnersh, Wokingham
Berkshire RG11 5AR
Telephone: (0734) 784 774

International Service Information

Not all Hewlett-Packard service centers offer service for all models of HP calculator products. However, if you bought your product from an authorized Hewlett-Packard dealer, you can be sure that service is available in the country where you bought it.

If you happen to be outside of the country where you bought your unit, you can contact the local Hewlett-Packard service center to see if service is available for it. If service is unavailable, please ship the unit to the address listed above under Obtaining Repair Service

in the United States. A list of service centers for other countries can be obtained by writing to that address.

All shipping, reimportation arrangements, and customs costs are your responsibility.

Service Repair Charge

There is a standard repair charge for out-of-warranty repairs. The repair charges include all labor and materials. In the United States, the full charge is subject to the customer's local sales tax. In European countries, the full charge is subject to Value Added Tax (VAT) and similar taxes wherever applicable. All such taxes will appear as separate items on invoiced amounts.

Calculator products damaged by accident or misuse are not covered by the fixed repair charges. In these situations, repair charges will be individually determined based on time and material.

Service Warranty

Any out-of-warranty repairs are warranted against defects in materials and workmanship for a period of 90 days from date of service.

Shipping Instructions

Should your unit require service, return it with the following items:

A completed Service Card, including a description of the problem.

A sales receipt or other documentary proof of purchase date if the one-year warranty has not expired.

The product, the Service Card, a brief description of the problem, and (if required) the proof of purchase date should be packaged in the original shipping case or other adequate protective packaging to prevent in-transit damage. Such damage is not covered by the one-year limited warranty; Hewlett-Packard suggests that you insure the shipment to the service center. The packaged unit should be shipped to the nearest Hewlett-Packard designated collection point or service center. Contact your dealer for assistance. (If you are not in the country where you originally purchased the unit, refer to International Service Information above.)

Whether the unit is under warranty or not, it is your responsibility to pay shipping charges for delivery to the Hewlett-Packard service center.

After warranty repairs are completed, the service center returns the unit with postage prepaid. On out-of-warranty repairs in the United States and some other countries, the unit is returned C.O.D. (covering shipping costs and the service charge).

Further Information

Service contracts are available. For information about service contracts, please contact the Calculator Service Center in Corvallis, Oregon.

Calculator product circuitry and design are proprietary to Hewlett-Packard, and service manuals are not available to customers.

Should other problems or questions arise regarding repairs, please call your nearest Hewlett-Packard service center.

When You Need Help

Technical Assistance. For technical assistance with this product, call:

(503) 757-2004
8 a.m. to 3 p.m.
Pacific time

or write to:

Hewlett-Packard Co.
Portable Computer Division
Calculator Technical Support
1000 N.E. Circle Blvd.
Corvallis, OR 97330

Product Information. For information about Hewlett-Packard products and prices, contact your local Hewlett-Packard dealer. For the name of the dealer nearest you, or to order free literature about Hewlett-Packard products, call toll-free:

(800) FOR-HPPC
(800) 367-4772

or write to:

Hewlett-Packard Co.
Personal Computer Group
PCG Telemarketing
10520 Ridgeview Court
Cupertino, CA 95014

Temperature Specifications

- Operating: 0° to 55° C (32° to 131° F)
- Storage: −40° to 65° C (−40° to 149° F)

Potential for Radio and Television Interference (for U.S.A. Only)

The HP-15C generates and uses radio frequency energy and if not installed and used properly, that is, in strict accordance with the manufacturer's instructions, may cause interference to radio and television reception. It has been type tested and found to comply with the limits for a Class B computing device in accordance with the specifications in Subpart J of Part 15 of FCC Rules, which are designed to provide reasonable protection against such interference in a residential installation. However, there is no guarantee that interference will not occur in a particular installation. If your HP-15C does cause interference to radio or television reception, you are encouraged to try to correct the interference by one or more of the following measures:

- Reorient the receiving antenna.
- Relocate the calculator with respect to the receiver.
- Move the calculator away from the receiver.

If necessary, you should consult your dealer or an experienced radio/television technician for additional suggestions. You may find the following booklet prepared by the Federal Communications Commission helpful: *How to Identify and Resolve Radio-TV Interference Problems*. This booklet is available from the U.S. Government Printing Office, Washington, D.C. 20402, Stock No. 004-000-00345-4.

Function Summary and Index

ON 272
Complex Functions 272
Conversions 273
Digit Entry 273
Display Control 273
Hyperbolic Functions 274
Index Register Control 274
Logarithmic and Exponential Functions 274
Mathematics 274
Matrix Functions 275
Number Alteration 276
Percentage 276
Prefix Keys 276
Probability 276
Stack Manipulation 277
Statistics 277
Storage 278
Trigonometry 278

ON

ON Turns the calculator's display on and off **(page 18)**. It is also used in resetting Continuous Memory **(page 63)**, changing the digit separator **(page 61)**, and in various tests of the calculator's operation **(pages 263-264)**.

Complex Functions

Re≷Im Real exchange imaginary. Activates Complex mode (establishing an imaginary stack) and exchanges the real and imaginary X-registers **(page 124)**.

I Used to enter complex numbers. Activates Complex mode (establishing an imaginary stack) **(page 121)**. Also used with DIM to indirectly dimension matrices **(page 174)**. (For Index register functions, refer to Index Register Control keys, page 274.)

[(i)] Displays the contents of the imaginary X-register while the key is held **(page 124)**.

[SF] 8 Sets flag 8, which activates Complex mode **(page 121)**.

[CF] 8 Clears flag 8, deactivating Complex mode **(page 121)**.

Conversions

[→R] Converts polar magnitude r and angle θ in X- and Y-registers respectively to rectangular x- and y-coordinates **(page 31)**. For operation in Complex mode, refer to page 134.

[→P] Converts x, y rectangular coordinates placed in X- and Y-registers respectively to polar magnitude r and angle θ **(page 30)**. For operation in Complex mode, refer to page 134.

[→H.MS] Converts decimal hours (or degrees) to hours, minutes, seconds (or degrees, minutes, seconds) **(page 27)**.

[→H] Converts hours, minutes, seconds (or degrees, minutes, seconds) to decimal hours (or degrees) **(page 27)**.

[→RAD] Converts degrees to radians **(page 27)**.

[→DEG] Converts radians to degrees **(page 27)**.

Digit Entry

[ENTER] Enters a copy of number in X-register (display) into Y-register; used to separate multiple number entries **(pages 22, 37)**.

[CHS] Changes sign of number or exponent of 10 in display **(pages 19, 124)**.

[EEX] Enter exponent; next digits keyed in are exponents of 10 **(page 19)**.

[0] through [9] digit keys **(page 22)**.

[·] Decimal point **(page 22)**.

Display Control

[FIX] Selects fixed point display mode **(page 58)**.

[SCI] Selects scientific notation display mode **(page 58)**.

[ENG] Selects engineering notation display mode **(page 59)**.

Mantissa. Pressing [f] CLEAR [PREFIX] displays all 10 digits of the number in the X-register as long as the [PREFIX] key is held down **(page 60)**. It also clears any partial key sequences **(page 19)**.

Hyperbolic Functions

[HYP] [SIN] [HYP] [COS] [HYP] [TAN] Compute hyperbolic sine, hyperbolic cosine, or hyperbolic tangent, respectively **(page 28)**.

[HYP^{-1}] [SIN], [HYP^{-1}] [COS], [HYP^{-1}] [TAN] Compute inverse hyperbolic sine, inverse hyperbolic cosine, or inverse hyperbolic tangent, respectively **(page 28)**.

Index Register Control

[I] Index register (R_I). Storage register for: indirect program execution—branching with [GTO] and [GSB], looping with [ISG] and [DSE]—indirect flag control, and indirect display format control **(page 107)**. Also used to enter complex numbers and activate Complex mode **(page 121)**.

[(i)] Indirect operations. Used to address *another* storage register *through* R_I for purposes of storage, recall, storage, arithmetic, and program loop control **(page 107)**. Also used with [DIM] to allocate storage registers **(page 215)**.

Logarithmic and Exponential Functions

[LN] Computes natural logarithm **(page 28)**.

[e^x] Natural antilogarithm. Raises e to power of number in display (X-register) **(page 28)**.

[LOG] Computes common logarithm (base 10) **(page 28)**.

[10^x] Common antilogarithm. Raises 10 to power of number in display (X-register) **(page 28)**.

[y^x] Raises number in Y-register to power of number in display (X-register) (enter y, then x). Causes the stack to drop **(page 29)**.

Mathematics

[−] [+] [×] [÷] Arithmetic operators; cause the stack to drop **(page 29)**.

[$\sqrt{x}$] Computes square root x **(page 25)**.

[x^2] Computes the square of x **(page 25)**.

[$x!$] Calculates the factorial ($n!$) of x or Gamma function (Γ) of $(1 + x)$ **(page 25)**.

[$1/x$] Computes reciprocal **(page 25)**. (For matrix use, refer to Matrix Functions, page 275.)

[π] Places value of π in display **(page 24)**.

[SOLVE] Solves for real root of a function $f(x)$, with the expression for $f(x)$ defined by the user in a labeled subroutine **(page 180)**.

[$\int$] Integrate. Computes the definite integral of $f(x)$, with the expression $f(x)$ defined by the user in a labeled subroutine **(page 194)**.

Matrix Functions

[DIM] Dimensions a matrix of a given name {[A] to [E], [I]} **(page 141)**.

[RESULT] Designates the matrix into which the result of certain matrix operations is placed **(page 148)**.

[USER] User mode. Row and column numbers in R_0 and R_1 are automatically incremented each time [STO] or [RCL] {[A] to [E], [(i)]} is pressed **(page 144)**.

[STO] and [RCL] {[A] to [E], [(i)]} Stores or recalls matrix elements using the row and column numbers in R_0 and R_1 **(pages 144, 146)**.

[STO] [g] and [RCL] [g] {[A] to [E], [(i)]} Stores or recalls matrix elements using the row and column numbers in the Y- and X-registers **(page 146)**.

[STO] and [RCL] [MATRIX] {[A] to [E]} Stores or recalls matrices for the specified matrix **(pages 142, 147)**.

[STO] and [RCL] [RESULT] Stores or recalls descriptor of the result matrix **(page 148)**.

[RCL] [DIM] {[A] through [E], [I]} Recalls the dimensions of the given matrix into the Y- (row) and X-(column) registers **(page 142)**.

[1/x] Inverts the matrix whose descriptor is displayed and places the result in the specified result matrix. The descriptor of the result matrix is then displayed **(page 150)**.

[+], [−], [×] Adds, subtracts, or multiplies the corresponding elements of two matrices or of one matrix and a scalar. Stores in result matrix **(page 152-155)**.

[÷] For two matrices, multiplies inverse of matrix in X by matrix in Y. For only one matrix: if matrix in Y, divides all elements of matrix by scalar in X; if matrix in X, multiplies each element of inverse of matrix by the scalar in Y. Stores in result matrix **(pages 152-155)**.

[CHS] Changes sign of all elements in matrix specified in X-register **(page 150)**.

[MATRIX] {0 through 9} Matrix operations.

[MATRIX] 0 Dimensions all matrices to 0×0 **(page 143)**.

[MATRIX] 1 Sets row and column numbers in R_0 and R_1 to 1 **(page 143)**.

[MATRIX] 2 Complex transform: $\mathbf{Z}^P$ to $\tilde{\mathbf{Z}}$ **(page 164)**.

[MATRIX] 3 Inverse complex transform: $\tilde{\mathbf{Z}}$ to $\mathbf{Z}^P$ **(page 164)**.

[MATRIX] 4 Transpose: $\mathbf{X}$ to $\mathbf{X}^T$ **(page 150)**.

[MATRIX] 5 Transpose multiply: $\mathbf{Y}$ and $\mathbf{X}$ to $\mathbf{Y}^T\mathbf{X}$ **(page 154)**.

[MATRIX] 6 Calculates residuals in result matrix **(page 159)**.

[MATRIX] 7 Calculates row norm of matrix specified in X-register **(page 150)**.

[MATRIX] 8 Calculates Frobenius norm of matrix specified in X-register **(page 150)**.

[MATRIX] 9 Calculates determinant of matrix specified in X-register (also does *LU* decomposition of the matrix) **(page 150)**.

[Cy,x] Transforms matrix stored in "partitioned form" ($\mathbf{Z}^P$) to "complex form" ($\mathbf{Z}^C$) **(page 162)**.

[Py,x] Transforms matrix stored in "complex form" ($\mathbf{Z}^C$) to "partitioned form" ($\mathbf{Z}^P$) **(page 162)**.

[x=0] [TEST] 0 [TEST] 5 [TEST] 6 Conditional tests for matrix descriptors in the X- or X- and Y-registers. [x=0] and [TEST] 0 ($x \neq 0$) test the quantity in the X-register for zero. Matrix descriptors are considered nonzero. [TEST] 5 ($x = y$) and [TEST] 6 ($x \neq y$) test if the descriptors in X and Y are the same. The result affects program execution: skip (one line) if false **(page 174)**.

Number Alteration

[ABS] Yields absolute value of number in display **(page 24)**.

[FRAC] Leaves only fractional portion of number in display (X-register) by truncating integer portion **(page 24)**.

[INT] Leaves only integer portion of number in display (X-register) by truncating fractional portion **(page 24)**.

[RND] Rounds mantissa of entire (10-digit) number in X-register to match display format **(page 24)**.

Percentage

[%] Percent. Computes *x* % (value in display) of number in the Y-register **(page 29)**. Unlike most two-number functions, [%] does not drop the stack.

[Δ%] Percent difference. Computes percent of change between number in Y-register and number in display **(page 30)**. Does not drop the stack.

Prefix Keys

[f] Pressed before a function key to select the gold function printed above that key **(page 18)**.

[g] Pressed before a function key to select the blue function printed below that key **(page 18)**.

For other prefix keys, refer to Display Control keys (page 273), Storage keys (page 278), and the Programming Summary and Index (page 278).

CLEAR [PREFIX] Cancels any prefix keystrokes and partially entered instructions such as [f] [SCI] **(page 19)**. Also displays the complete 10-digit mantissa of the number in the display **(page 60)**.

Probability

[Cy,x] Combination. Computes the num-

ber of possible sets of *y* different items taken *x* at a time, and causes the stack to drop **(page 47)**. (For matrix use, refer to Matrix Functions keys, page 276.)

[P y,x] Permutation. Computes the number of possible different arrangements of *y* different items taken *x* at a time, and causes the stack to drop **(page 47)**. (For matrix use, refer to Matrix Functions keys, page 276.)

Stack Manipulation

[x≷y] Exchanges contents of X- and Y-stack registers **(page 34)**.

[x≷] X-register exchange. Exchanges contents of X-register with those of any other named storage register. Used with [I], [(i)], digit, or [.] digit address **(page 42)**.

[Re≷Im] Real exchange imaginary. Exchanges the contents of the real and imaginary X-registers and activates Complex mode **(page 124)**.

[R↓] Rolls down contents of stack **(page 34)**.

[R↑] Rolls up contents of stack **(page 34)**.

[CLx] Clears contents of display (X-register) to zero **(page 21)**.

[←] In Run mode: removes the last digit keyed in, or clears the display (if digit entry has been terminated) **(page 21)**.

Statistics

[Σ+] Accumulates numbers from X- and Y-registers into storage registers R_2 through R_7 **(page 49)**.

[Σ−] Removes numbers in X- and Y-registers from storage registers R_2 through R_7 for correcting [Σ+] accumulations **(page 52)**.

[x̄] Computes mean of *x*- and *y*-values accumulated by [Σ+] **(page 53)**.

[s] Computes sample standard deviations of *x*- and *y*-values accumulated by [Σ+] **(page 53)**.

[ŷ,r] Linear estimate and correlation coefficient. Computes estimated value of *y* ($\hat{y}$) for a given value of *x* by least squares method and places result in X-register. Computes the correlation coefficient, *r*, of the accumulated data and places result in Y-register **(page 55)**.

[L.R.] Linear Regression. Computes the *y*-intercept and slope for the linear function best approximating the accumulated data. The value of the *y*-intercept is placed in the X-register; the value of the slope is placed in the Y-register **(page 54)**.

[RAN#] Random number. Yields a pseudo-random number as generated from a seed stored using [STO] [RAN#] **(page 48)**.

CLEAR [Σ] Clears contents of the statistics registers (R_2 to R_7) **(page 49)**.

Storage

[STO] Store. Stores a copy of a number into the storage register specified {0 to 9, .0 to .9, [I], [(i)]} **(page 42)**. Also used for storage register arithmetic: new register contents = old register contents { [+], [−], [×], [÷] } display **(page 44)**.

[RCL] Recall. Recalls a copy of the number from the storage register specified {0 to 9, .0 to .9, [I], [(i)]} **(page 42)**. Also used for storage register arithmetic: new display = old display { [+], [−], [×], [÷] } register contents **(page 44)**.

CLEAR [REG] Clears contents of all storage registers to zero **(page 43)**.

[LSTx] Recalls into the display the number present before the previous operation **(page 35)**.

Trigonometry

[DEG] Sets decimal Degrees mode for trigonometric functions—indicated by absence of **GRAD** or **RAD** annunciator **(page 26)**. Not operative for complex trigonometry.

[RAD] Sets Radians mode for trigonometric functions—indicated by **RAD** annunciator **(page 26)**.

[GRD] Sets Grads mode for trigonometric functions—indicated by **GRAD** annunciator **(page 26)**. Not operative for complex trigonometry.

[SIN], [COS], [TAN] Compute sine, cosine, or tangent, respectively, of number in display (X-register) **(page 26)**.

[SIN^{-1}], [COS^{-1}], [TAN^{-1}] Compute arc sine, arc cosine, or arc tangent, respectively, of number in display (X-register) **(page 26)**.

Programming Summary and Index

[P/R] Program/Run mode. Sets the calculator to Program mode (**PRGM** annunciator on) or Run mode (**PRGM** annunciator cleared) **(page 66)**.

CLEAR [PRGM] In Program mode: clears all program memory and deallocates all program registers. In Run mode: only resets calculator to line 000 **(page 67)**.

[MEM] Displays current status of calculator memory (number of registers dedicated to data storage, the common pool, and program memory) **(page 215)**.

[←] Back arrow. In Program mode, deletes displayed instruction from program memory. All subsequent instructions are moved up **(page 83)**.

[LBL] Label. Used with the label designations below to denote the start of a program routine **(page 67)**.

[A] [B] [C] [D] [E] 0 1 2 3 4 5 6 7 8 9 .0 .1 .2 .3 .4 .5 .6 .7 .8 .9 Label designations. When preceded by [LBL], define the beginning of a program routine **(page 67)**. Also used (without [LBL]) to initiate execution of a specific routine **(page 69)**.

[USER] Activates and deactivates User mode, which exchanges the primary (white) and gold alternate functions ([A] through [E]) of the top left five functions **(page 69)**. User mode also affects the matrix use of [STO] or [RCL] {[A] through [E], [(i)]}. User mode automatically increments R_0 (row number) or R_1 (column number) for storage or recall of matrix elements **(page 144)**.

[GTO] Go to. Used with a label designator (listed above) or [I] to transfer the position of the calculator to the designated label. If it is a program instruction, program execution continues. If it is not a program instruction, only the position change occurs **(page 90)**. If a negative number is stored in R_I, [GTO] [I] will effect a transfer to a *line number* **(page 109)**.

[GTO] [CHS] *nnn* Go to line number. Positions calculator to the existing line number specified by *nnn*. Not programmable **(page 82)**.

[GSB] Go to subroutine. Used with a label designator (listed above) or to start the execution of a given, labeled routine. Can be used both in a program and from the keyboard (in Run mode). A [RTN] instruction transfers execution back to the first line following the [GSB] **(page 101)**.

[BST] Back step. Moves calculator back one or more lines in program memory. (Also scrolls in Program mode.) Displays line number and contents of previous program line **(page 83)**.

[SST] Single step. In Program mode: moves calculator forward one or more lines in program memory. In Run mode: displays and executes the current program line, then steps to next line to be executed **(page 82)**.

[PSE] Pause. Halts program execution for about 1 second to display contents of X-register, then resumes execution **(page 68)**.

[R/S] Run/Stop. Begins program execution from current line number in program memory. Stops execution if program is running **(page 68)**.

[RTN] Return. Causes calculator to return to line 000 and halt execution (if running) **(page 68)**. If in a subroutine, merely returns to line after [GSB] **(page 101)**.

[SF] Set flag (= true). Sets designated flag (0 to 9). Flags 0 through 7 are user flags, flag 8 signifies Complex mode, and flag 9 signifies an overflow condition **(page 92)**.

[CF] Clear flag (= false). Clears designated flag (0 to 9) **(page 92)**.

[F?] Is flag set? Tests for designated flag. If set, program execution continues; if cleared, program execution skips one line before continuing **(page 92)**.

[x≤y] [x=0] [TEST] {0 through 9} Conditional tests. Each test compares value in X-register against 0 or value in Y-register as indicated. If true, calculator executes instruction in next line of program memory. If false, calculator skips one line in program memory before resuming execution **(page 91)**. [x=0] and [TEST] 0, 5, and 6 are also valid for complex numbers and matrix descriptors **(pages 132, 174)**.

[TEST] 0 $x \neq 0$
[TEST] 1 $x > 0$
[TEST] 2 $x < 0$
[TEST] 3 $x \geq 0$
[TEST] 4 $x \leq 0$
[TEST] 5 $x = y$
[TEST] 6 $x \neq y$
[TEST] 7 $x > y$
[TEST] 8 $x < y$
[TEST] 9 $x \geq y$

[DSE] Decrement and skip if equal to or less than. Decrements counter value in given register as stipulated. Skips one program line if new counter value is equal to or less than specified test value **(page 109)**.

[ISG] Increment and skip if greater than. Increments counter value in given register as stipulated. Skips one program line if new counter value is greater than specified test value **(page 109)**.

Subject Index

Page numbers in **bold** type indicate primary references; page numbers in regular type indicate secondary references.

A

Abbreviated key sequences, **78**
Absolute value (ABS), **24**
Allocating memory, 42, **213-219**
Altering program lines, **83**
Annunciators,
  complex, **121**
  list of, **60**
  PRGM, 32, **66**
  trigonometric, **26**
Antilogarithms, common and natural, **28**
Arithmetic operation, **29**, 37
Assistance, technical, **270**
Asymptotes, horizontal, **230**
Automatic incrementing of row and column numbers, **143**

B

Back-stepping (BST), **83**
Bacterial population example, **41**
Battery life, **259**
Battery replacement, 260, **261-263**
Bessel functions, **195**, 197
Branching,
  conditional, **91**, 98, 177, 192
  indirect, **108-109**, 112-114, 115
  simple, **90**

C

C annunciator, 99, **121**
Can volume and area example, **70-74**
Chain calculations, **22-23**, 38

Changing signs, **19**
in Complex mode, **124-125**
in matrices, **177**
[CHS], **19**
Clearing
blinking in display, **100**
complex numbers, **125-127**
display, **21**
operations, **20-21**
overflow condition, 45, **61**
prefix keys, **19**
statistics registers, **49**
Coefficient matrix, **156**
Combinations function ([Cy,x]), **47**
Common pool, **213**
Complex arithmetic example, **132**
Complex conjugate, forming, **125**
Complex matrix,
inverting, 162, 164, **165**
multiplying, 162, 164, **166**
storing elements, **161**
transforming, **162, 164**
Complex mode, **120-121**
activating, 99, **120-121**, 133
deactivating, **121**
mathematics functions in, **131**
stack lift in, **124**
Complex numbers,
clearing, **125-127**
converting polar and rectangular forms, **133-135**
entering, **121**, 127, 128-129
storing and recalling, **130**
Conditionals, indirect, **109-111**, 112, 116
Conditional tests, **91**, 98, 192
in Complex mode, **132**
with matrix descriptors, **174**
Constant matrix, **156**
Constants,
calculations with, **39-42**
using in arithmetic calculations, 35, **39-42**
Continuous Memory,
duration of, **62**

resetting (clearing), **63**
what it retains, 43, 48, 58, 61, **62**
Conventions, handbook, **18**
Conversions,
degrees and radians, **27**
polar and rectangular coordinates, **30-31**
time and angle, **26-27**
Correcting accumulated statistics data, **52**
Correlation coefficient, find the ([ŷ,r]), **55-56**
[COS], [COS⁻¹], **26**
Counters in program loops, 98, **112-114**
Crocus example, **43**
Cumulative calculations, **41**

D

Data storage, **42**
Data storage pool, **213-214**
Debt payment example, **95**
Decimal point, 22
Decimal point display, **61**
Deflation, **233**, 234, 237
[DEG], **26**
Determinant, 150
Digit entry, 22
in Complex mode, **121**, 125, 127, **128-129**
termination, **22**, 36, **209**
Digit separator display, **61**
[DIM], 76-77, **215-217**
Disabling stack lift, **36**
Display (*See also* X-register),
blinking, **100**
clearing, **21**
error messages, **61**
full mantissa, **60**
in Complex mode, **121**
Display format, **58-59**, 61
effect on [f], **200**, 241, 244, **245-249**
Do if True rule, **92**, 192
[DSE], **109-111**, 112, 116

E

[EEX], **19**
Electrical circuit example, **169-171**
Enabling stack lift, **36**
[ENG], **59**
Engineering notation, **59**
[ENTER], 12, **33-34**, 36
  effect on digit entry, **22**, 29
  effect on stack movement, **37**, 41
Entering data for statistical analysis, **49**
Error
  conditions, **205-208**
  display, **61**
  stops, **78**
Errors,
  with [f], **203-204**
  with [SOLVE], **187**, 192, 193
Euclidean norm (*See* Frobenius norm)
Exchanging the real and imaginary stacks, **124**
Exponential function (*See* Power function)
Exponents, **19**, **20**

F

[f], **18**
Factorial function ([x!]), **25**
Falling stone example, **14**
[FIX], **58**
Fixed decimal notation, **58**
Flag tests, **92**, 98
Flag 8, **99**
Flag 9, **100**
Format, handbook, 2, 18
Fractional portion ([FRAC]), **24**
Frobenius norm, 150, **177**
Functions, nonprogrammable, **80**
Functions, one-number, 22, **25**
Functions, primary and alternate, **18**
Functions, two-number, 22, **29**

G

[g], **18**

Gamma function ([x!]), **25**
[GRD], **26**
[GSB], **101**
[GTO], **90**, 97, 98
[GTO] [CHS], **82**

H

Horner's Method, **79**, 181
Hyperbolic functions, **28**

I

Imaginary stack,
  clearing the, **124**
  creation of, **121-123**, 133
  display of, **124**
  stack lift of, **124**
Index register
  arithmetic, **108**, 112
  display format control, **109**, 114, 115, 116
  exchange with X-register, **108**, 112
  flag control, **109**, 115
  loop control, 107, **109-111**
  storage and recall, **107**, 111, 115
Indirect addressing, **106-108**, 115
Initialization, **87**
Instructions, **74**
Integer portion ([INT]), **24**
Integrate function ([∫]), **194-204**
  accuracy of, **200-203**, 240, **241-245**
  algorithm for, 196, **240-241**, 249-251, 255-256
  display format with, **245-249**
  execution time for, 196, **200**, 244, 245, **254-256**
  memory usage, **204**
  obtaining an approximation for, **257-258**
  problems with erratic functions, **249-254**
  programmed, **203-204**
  recursive use of, **203**
  subroutines for, **194-195**
  uncertainty in, **202-203**, 240-244, **245-249**
Interchanging functions (*See* User mode)
Interference, radio and television, **271**

Intermediate results, **22**, 38
Interpolation, using [ŷ,r], **57**
[ISG], **109-111**, 116
Iterations using [ISG] and [DSE], **111**

K

Keycodes, **74-75**
Keying in
  chain calculations, **22**
  exponents, **19-20**
  one-number functions, **22**
  two-number functions, **22**, 29

L

Labels, **67**, **77**, 90, 97
LAST X register, **35**
  in matrix functions, **174-176**
  operations saved by, **212**
  putting constants in, **39-40**
  to correct statistics data, **52**
Linear equations, solving with matrices, **138**, 156
Linear estimation ([ŷ,r]), **55-56**
Linear regression ([L.R.]), **54**
Loading the stack with constants, 39, **41**
Logarithmic functions, common and natural, **28**
Loop control number, **109**, 116
Looping, **90**, 98
Low-power indication, 62, **260-261**
LU decomposition, **148**, 155, 156, 160
Łukasiewicz, Jan, **32**

M

Mantissa, displaying full 10 digits, **60**
Matrix
  complex, **160-163**
  copying, **149**
  descriptors, 139, **147**, 160, in R_I, **173-174**
  dimensioning, **140**, 142, 142, 174
  dimensions, displaying, **142**,147
  equation, complex, **168**
  memory, **140**, 171

name (*See* Matrix descriptors)
partitioned, **161**, 164
Matrix elements,
accessing individually, **145-147**
displaying, **144**
storing and recalling, **143-144**, 147, 149, 176
Matrix functions,
using R_I, **173-174**
using registers, **173**
arithmetic, **153**
conditional, **177**
inverse, **150**, 154
multiplication **154**
one-matrix, **149-151**
programmed, **176-177**
reciprocal, 150
residual, **159**
row norm, 150, **177**
summary, **177-179**
transpose, **150**, 151, 154
Mean ($\bar{x}$), **53**
[MEM], **215**
Memory
allocation, 76, **215-217**
availability, 75-77, 213, **215**
configuration, initial, **75-76**
distribution, 75, **213-214**
limitations, 75, 77, **217**
requirements for advanced functions, **218-219**
requirements for programming, **218**
stack (*See* Stack)
status display, **215**
registers in, **213-215**
Metal box dimensions example, **189-191**
Minima, finding with [SOLVE], **230**
Modes, trigonometric, **26**
Multiple roots, **234**

N

Negative numbers, **19**
in Complex mode, **124-125**

Nested calculations, **38**
Neutral operations, **211**
Nonprogrammable functions, **80**
Normalizing statistics data, **50**
null display, **144**, 149

O

[ON],
  and off, **18**
  to reset Continuous Memory, **63**
  to set decimal point display, **61**
Overflow condition, 45, **61**, 100

P

[P/R], **66**, **68**
Pause ([PSE]), **68**
Percent difference ([Δ%]), **29**
Percentage functions, **29-30**
Permutations function ([Py,x]), **47**
Phasor notation, 133
Pi, **24**
Polar coordinates, **30**, in Complex mode, **133-135**
Power function ([y^x]), **29**
Prefix keys, **19**
PRGM annunciator, **66**, **82**
Program
  control, indirect, 107, **109-111**
  data entry techniques, **69-70**
  end, **68**, 77
  entering, **66-68**
  labels, **67**, 77
  loading **66**
  loop counters, **109**, 112-114, 116
  mode, **66**, **68**, 86
  position, changing, **82**, 86
  running, **68-69**
  starting, **69**
  stops, **68**, 78
Program execution, **69**
  after [GSB], **101**
  after [GTO], **97**

after overflow, **100**
after test, **92**
from or through labels, **78-79**
Program lines (instructions), **67**, 74
deleting, **83**, 86
inserting, **83**, 86
Program memory, 67, 70, **75**, **217-219**
automatic reallocation, **217-218**
clearing, **67**
moving in, **67**

Q

Quadratic equation, solving, **181**

R

R_0 and R_1, using to access matrix elements, 143, **146**, 176
[RAD], **26**
Radioisotope example, **93-94**
Random number generator ([RAN#]), **48**
Random number storage and recall, **48**
Recall arithmetic, **44**
Recalling accumulated statistics data, **50**
Recalling numbers ([RCL]), **42**, 44, with matrices, 144, 149, 176
Reciprocal ([1/x]), **25**, with matrix, 150
Rectangular coordinates, **31**, in Complex mode, **133-135**
Registers, converting, **215-217**
Residual, **159**
Result matrix, 147, **148**, 150, 152
Return ([RTN]), **68**, **77**
Returns, pending, **101**, 105, 192, 204
Reverse Polish Notation, **32**
[Re≷Im], **124**, 127
Rice yield example, **50-56**
Ridget hurling example, **184-186**, 224-226
Roll down, **34**
Roll up, **34**
Roots, eliminating, **233**, 234, 237
Roots, meaningless, **188**, 191
Rounding ([RND]), **24**
Rounding in the display, **59**
Round-off errors, 52, **60**, with [SOLVE], **223**, 237

Row norm, 150, **177**
Run/Stop (R/S), **68**, 91
running display, **69**, 147, 182

S

Scalar operations, **151-153**
SCI , **58**
Scientific notation, **58**
Scrolling, **82**
Secant line calculation example, **102**
Self-tests, **263-265**
Service information, **267-270**
Shear stress example, **227-228**
SIN , SIN^{-1} , **26**
Sine integral example, **198-199**
Single-stepping (SST), **82**, 85
Skip if True rule, **110**
Slope, finding the, **54**
SOLVE , **180-181**
  accuracy, **222-226**, specifying, **238**
  algorithm, 182, 187-188, **220-222**, 230-231
  conditions necessary for, **221-222**
  constant function value with, **187**, 189
  execution time, **238**
  illegal math routine with, **187-188**
  initial estimates with, 181, **188-192**, 221, 233, 237
  memory usage, **193**
  nonzero minimum of function with, **187**
  programmed, **192**
  recursive use of, **193**
  restrictions on, **193**
  using as a conditional test, **192**
  using functions with discontinuities, **227**
  using functions with poles, **227**
  using functions with several roots, **233-238**
  with no root, **186-188**, 192, 229
Square root ($\sqrt{x}$), **25**
Squaring (x^2), **25**
Stack
  contents, with f , 197, **202**
  drop, **33**, 38
  lift, **33**, 36, 38, 44, **209-211**

manipulation functions, **33-34**, in Complex mode, **131**
imaginary, **120-125**
used to access matrix elements, **146-147**
Stack-disabling operations, **210**
Stack-enabling operations, **210-211**
Stack movement, **32**, 33-37
in matrix functions, **174-176**
with [SOLVE], **181**
Standard deviation ([s]), **53**, sample vs. population, **53**
Star example, **40**
Statistics, accumulation of data ([Σ+]), **49**
Statistics, correction of accumulated data ([Σ-]), **52**
Statistics functions,
combinations, **47**
correlation coefficient, **55**
linear estimation, **55**
linear regression, **54**
mean, **53**
permutations, **47**
probability, **47**
standard deviation, **53**
Statistics registers, **49-50**
Status indicators, **60**
Storage and recall ([STO], [RCL]), **42**, 43, 44
complex numbers, **130**
direct (with [I]), 106, **107**
indirect, **106-107**, 111
matrices, **144**, 149, 176
matrix elements, **143-144**, 147, 149
Storage arithmetic, **43**
Storage registers, **42**
allocation, 42, **215-217**
arithmetic, **43**
clearing, **43**
statistics, 42, **49**
Subroutine
levels, **102**, 105
limits, **102**, 105
nesting example, **103**
returns, **101**, 105
using with [SOLVE], **180-181**, 192
System flags, 92, **99**

T

T-register, 32, **33**
  in matrix functions, **174-176**
  with [f], **202**
[TAN], [TAN⁻¹], **26**
Temperature specifications, **270**
[TEST], **91**
Tracing, 82
Transpose, **150**,151, 154
Trigonometric modes in Complex mode, **121**, 134
Trigonometric operations, **26**

U

u display, **176**
Uncommitted registers, 213, **215**, 217
Underflow,
  in any register, **61**
  storage register arithmetic, **45**
  with [SOLVE], **223**
User flags, **92**
User mode, 69, **79**, with matrices, 143, **176**

V

Vector arithmetic, using statistics functions, **57**

W

Warranty information, **265-267**
Wrapping, **86**, 90

X

X exchange ([x⇄]), **42**
X exchange Y ([x⇄y]), **34**
X-register, 32, 35, 37, 42, 60, **209-210**
  imaginary, **210**, **211**
  in matrix functions, 141, 156, **175-176**
  with [f], **202**
  with [SOLVE], 181, **183**, 192, 226

Y

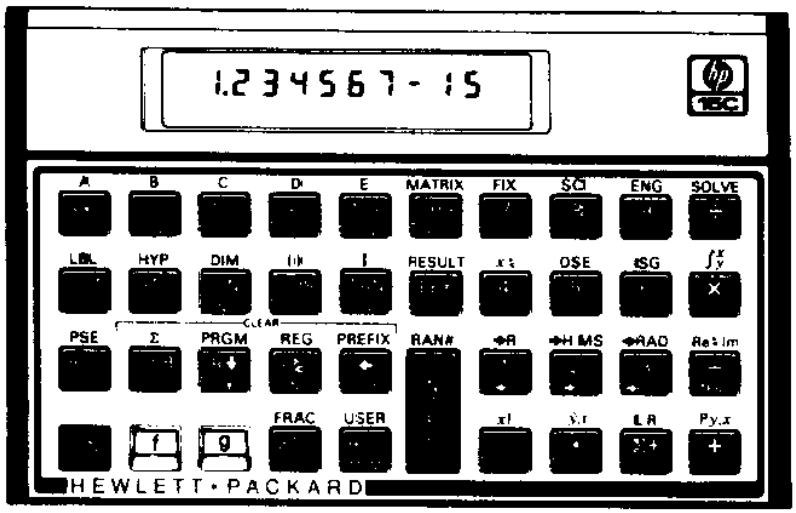
y-intercept, finding, **54**

Y-register, 32, 37
in matrix functions, 141, 156, **175-176**
with [f], **202**
with [SOLVE], 181, **183**, 192, 226

Z

Z-register, 32
in matrix functions, **174-176**
with [f], **202**
with [SOLVE], 181, **183**, 192, 226

The HP-15C
Keyboard and Continuous Memory



MEMORY STACK

	Real	Imaginary
T		
Z		
Y		
X		

Display shows real X-register.

LAST X

DATA STORAGE POOL

R_I

R_0	0	
R_1	1	
R_2	2	n
R_3	3	Σx
R_4	4	Σx^2
R_5	5	Σy
R_6	6	Σy^2
R_7	7	Σxy
R_8	8	
R_9	9	

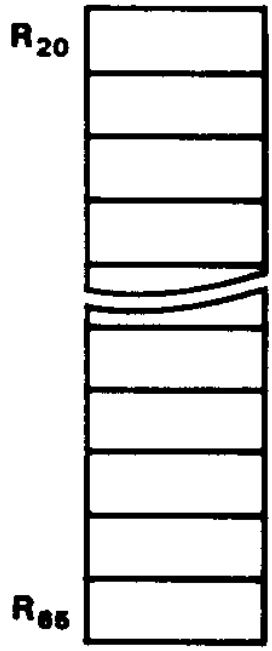
$R_{.0}$	10
$R_{.1}$	11
$R_{.2}$	12
$R_{.3}$	13
$R_{.4}$	14
$R_{.5}$	15
$R_{.6}$	16
$R_{.7}$	17
$R_{.8}$	18
$R_{.9}$	19

Initial allocation is R_0 through R_{19} to data storage.

Allocations can be changed with the [DIM] [(i)] function.

There are seven bytes of memory per register. One or two bytes are needed per program instruction. One register at a time is converted to program memory as needed, starting at the highest-numbered available register and proceeding to the lower registers.

COMMON POOL



Matrix Memory

Imaginary Stack

[SOLVE] and [∫]

Uncommitted Registers

Program Memory up to seven program lines per register

Memory allocation for program lines is automatic *within* the common memory pool.

Initial allocation is R_{20} through R_{65} to the common pool, from which the above functions and programming draw memory space.