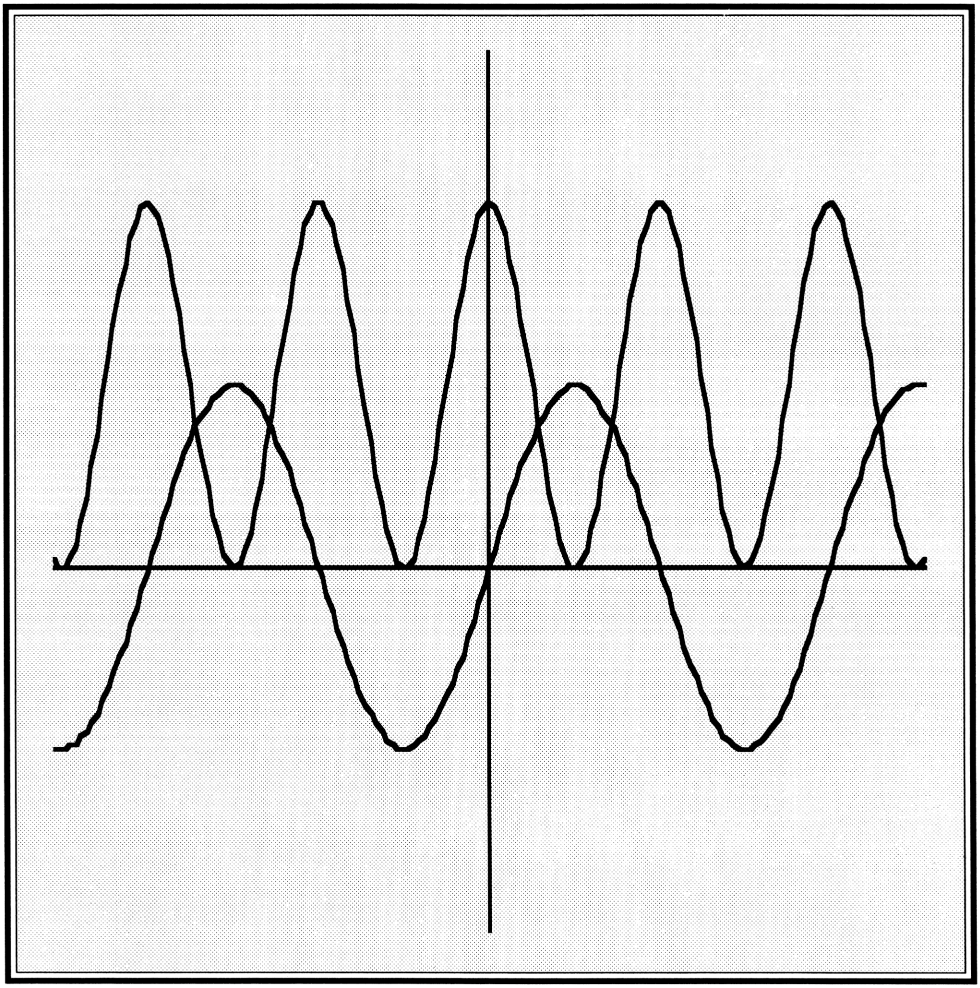
Calculus on the HP 48G/GX



By Dan Coffin

Calculus on the HP 48G/GX

by Dan Coffin

Grapevine Publications, Inc.
P.O. Box 2449
Corvallis, Oregon 97339-2449 U.S.A.

Acknowledgments

The terms "HP 48" and "48" are used for convenience herein to refer to the HP 48GX and HP 48G, the registered trade names for the handheld calculator/computer products of Hewlett-Packard Company. For allowing me to use and (sometimes) to modify their programs for inclusion here, many thanks are due to Joseph Horn (PFIT) and William Wickes (CHVAR, FDER, FUNOF?, INDEF, INTEVAL, LININ?, NSEGINT, →RPN, SCOF, SDET, SEGINT, SM→, →SM, and XINT). My thanks go also to Thomas Dick, Charles Patton, and the Oregon State University Calculus Curriculum Project for their inspiring approach to teaching introductory calculus.

Printed in the United States of America
ISBN 0-931011-44-2

First Printing – September, 1995

0. Start Here

What Is This Book?

This book is to help you use the HP 48G/GX to improve your understanding of the mathematical topics usually found in a course in first-year calculus. Calculus is the mathematics of change—such as motion, growth, or acceleration. It generalizes methods of algebra and geometry to allow useful computations of continuously changing systems *without assumptions about ideal shapes, static functions or average values*. Specifically, calculus has two complementary uses:

- It provides a means to "freeze" the moment—to analyze the rate a system is changing at a particular instant in time.
- It provides a means to extrapolate from knowledge about how a system is changing from moment to moment to how will have changed overall.

The first of these contributions is the domain of *differential calculus*; the second is that of *integral calculus*. These two domains are linked by the ability to mathematically cope with infinity—infinitely tiny intervals of time and infinitely long sums—using the concept of a *limit*.

Calculus on the HP 48G/GX is organized much like a standard text. Chapters are divided into topics; topics are divided into examples. The examples demonstrate how to use the HP 48 to solve problems in a typical Calculus course—and they sometimes use programs (listed in the Appendix) to allow easy repetition.

However, this book isn't meant to *replace* your textbook. It does not try to rigorously justify the techniques and concepts used in problem-solving. Also, there may be topics treated in greater depth in your textbook than in this book, or vice versa. (Indeed, you may also wish to read Grapevine's *Algebra and Pre-Calculus on the HP 48G/GX* to get additional tools and instruction on related topics.)

Before using this book, you should be able to do these things on the HP 48G/GX:

- Perform basic arithmetic and navigate the various menus;
- Enter, name, and use variables, lists, algebraic expressions, and programs.

If these concepts are still "fuzzy" for you, *stop here* and work through either the *Quick Start Guide* (which came with the calculator) or the first three chapters of Grapevine's *Easy Course* book on the HP 48G/GX.

1. Series, Sequences, and Limits

Introduction to Sequences and Series

A *sequence* is an infinitely long, ordered list of numbers. Each number is a *term* of the sequence, and its position within the list is its *index*. Usually a sequence has a *defining function* for which the index is the input and the term the output.

Associated with every sequence is a *series*—a single value—the sum of the terms of the sequence. Computing a series can be tricky because is an infinite sum—the sum of an infinite number of terms.

In a sense, therefore, a series is never computed in its entirety; after computing any *partial sum* there are always additional terms to be added. So, mathematically speaking, a series is the *limit* of the partial sums of the terms of a sequence. That is why a collective term, "series," is used for a singular value—a sum: a series implies the sequence (the collection) of partial sums of which it is the limit.

Series come in two flavors. Those with finite value are called *convergent*; those with infinite value are known as *divergent*. If the terms of the sequence approach zero as their indices increase, then it is possible that the series is convergent—a *finite* number as the limit of an *infinite* sum. However, if the terms of a sequence don't approach zero as their indices increase, then the series has an infinite value —it is divergent.

One of the important uses of convergent series is to approximate measurements and values that otherwise are difficult (even impossible) to compute otherwise.

You'll see each of these ideas developed in greater detail in this chapter, along with demonstrations on the relevant uses of the HP 48.

Sequences

Although any list of numbers might be a sequence, the important sequences are those lists of numbers that have defining rules to generate each term. There are two kinds of defining rules that create sequences in two different ways:

Closed-form sequences are formed by defining rules that are functions of the index: you input the index number and compute the value of the term. For example, the sequence { 0, 1, 3, 7, 15, 24, 35, 48, ... } has a closed-form defining rule: $a_n = n^2 - 1$, where a_n is the term and n is its index.

Closed-form sequences are generally represented as $\{f(n)\}_{n=j}^{k}$, where $f(n)$ is the defining function, n is the index, j is the starting index value, and k is the ending index value. There are several important closed-form sequences:

- The terms of the harmonic sequence are reciprocals of the positive integers:
$$a_n = \frac{1}{n}$$
- An arithmetic sequence is a closed-form sequence where the *difference* of any two consecutive terms is constant.
- A geometric sequence is a closed-form sequence where the *ratio* of any two consecutive terms is constant.

Recursive sequences are formed by defining rules that are functions of the previous term. These rules define the first term of the sequence and use a recursion formula to compute the other terms of the sequence: input one or more previous terms and compute the value of the next term. Recursive sequences require that you know previous terms before you can compute a new term.

For example, the sequence { 0, 3, 9, 21, 45, 93, 189, ... } has a recursive definition:

$$a_1 = 0$$
$$a_n = 2a_{n-1} + 3$$

On the HP 48, sequences are most naturally represented by lists of terms. But the machine is finite, so any such list will be, also. To represent infinite, closed-form sequences, therefore, you must use the symbolic capabilities of the HP 48.

Example: Create the closed-form sequence: $\left\{2^n - 1\right\}_{n=1}^{\infty}$ on the HP 48.

1. Enter the expression in symbolic form: ['][2][y^x][α][⬅][N][−][1][ENTER].
2. Enter the index variable (n, in this case): ['][α][⬅][N][ENTER].
3. Combine these two objects into a list: [PRG] LIST [2] →LIST.

   Result: { '2^n-1' n }

Example: Compute the 15th term of the sequence in the previous example.

1. Make a copy of the sequence, then disassemble it: [ENTER] OBJ→ [◆].
2. Enter the desired index value and store it in the index variable: [1][5] [ENTER][SWAP][STO].
3. Evaluate the closed-form function: [EVAL]. Result: 32767

Example: Find the first seven terms of the sequence in the previous example.

For closed-end sequences, you can use the built-in SEQ command to generate a finite list of terms.

1. With the sequence already on the stack from the previous example (or after re-entering it, if necessary), first disassemble the list: ([◆], if necessary) [PRG] LIST OBJ→ [◆].
2. Enter the starting and ending indexes: [1][ENTER][7][ENTER].
3. Enter the step interval. In this case you want each term without skipping any, so the step interval is one: [1][ENTER].
4. Generate the list by using the SEQ command: PROC [NXT] SEQ.

   Result: { 1 3 7 15 31 63 127 }

Using a recursive sequence on the HP 48 is more involved. The program →SEQ (see page 322) takes the defining function for the sequence from level 3, a list of values of one or more initial terms from level 2, and an index number, n, from level 1. If n is nonnegative, →SEQ returns the term of the sequence with the given index. If n is negative, it will return a list of values of all terms from a_0 through a_n. →SEQ assumes the variables named 'a1', 'a2', 'a3', etc., represent previous terms: 'a1' is the earliest previous term; 'a2' is the next previous term, etc.

The following examples assume that the program →SEQ is stored in the current directory path. (This assumption goes for all programs in this book—that they already correctly keyed in and stored in the current directory path prior to use.)

Example: Compute the 27th term in the sequence $\begin{aligned} a_1 &= 0 \\ a_n &= 2a_{n-1} + 3 \end{aligned}$

1. Enter the defining rule, using the 'a1', 'a2', ... convention as described above: ['] [2] [×] [α] [⬅] [A] [1] [+] [3] [ENTER].
2. Enter the values of the initial terms (there's only one in this case) in the form of a list: [⬅] [{ }] [0] [ENTER].
3. Enter the index for the term you wish to compute—as a positive number because you only want the term, not a list: [2] [7] [ENTER].
4. Execute →SEQ: [α] [α] [➡] [→] [S] [E] [Q] [ENTER] or [VAR] (then [NXT] or [⬅] [PREV] as needed) →SEQ. Result: 201326589.

Example: Find the first eight terms of the sequence described above.

1. Enter the defining rule: ['] [2] [×] [α] [⬅] [A] [1] [+] [3] [ENTER].
2. Enter the values of the initial terms as a list: [⬅] [{ }] [0] [ENTER].
3. Enter the number of terms you wish to compute—as a negative number because you want the full list of terms: [8] [+/-] [ENTER].
4. Execute →SEQ: [α] [α] [➡] [→] [S] [E] [Q] [ENTER] or [VAR] ([NXT] or [⬅] [PREV]), →SEQ. Result: { 0 3 9 21 45 93 189 381 }

As you can tell from the speed of computation, recursive rules are less efficient than closed-form defining rules for computing terms.

The *Fibonacci* sequence appears often in nature and has fascinating properties. Each term is the sum of the preceding two—obvious from its recursive form but not from its closed-end form. However, the closed form offers easier calculating.

Closed: $a_n = \frac{1}{\sqrt{5}}\left[\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n\right]$ *Recursive:* $a_1 = 1$, $a_2 = 1$, $a_n = a_{n-1} + a_{n-2}$

Example: Create the closed-form of the sequence and store it as 'FIBN'.

1. Enter the closed-form function of the Fibonacci: [⬅] [EQUATION] [1] [÷] [√x] [5] [▶] [▶] [⬅] [()] [⬅] [()] [▲] [1] [+] [√x] [5] [▶] [▶] [2] [▶] [▶] [y^x] [α] [⬅] [N] [▶] [−] [⬅] [()] [▲] [1] [−] [√x] [5] [▶] [▶] [2] [▶] [▶] [y^x] [α] [⬅] [N] [ENTER].
2. Enter the index variable: ['] [α] [⬅] [N] [ENTER].
3. Now combine these into a list and name it FIBN: [2] [PRG] LIST →LIST ['] [α] [α] [F] [I] [B] [N] [ENTER] [STO].

Example: Compute F_{18}, the 18th term, using the closed-form of the sequence.

1. Enter the Fibonacci sequence, stored in FIBN: [VAR] FIBN.
2. Disassemble the sequence list; enter the index value: [PRG] LIST OBJ→ [⬅] [1] [8] [ENTER].
3. Compute the value of the term: [SWAP] [STO] [EVAL]. Result: 2584

Example: Create a list of the terms of the Fibonacci sequence from F_{23} to F_{31}.

1. Enter the Fibonacci sequence, stored in FIBN: [VAR] FIBN.
2. Disassemble the sequence list: [PRG] LIST OBJ→ [⬅].
3. Enter the starting and ending index values: [2] [3] [ENTER] [3] [1] [ENTER].
4. Enter the step interval value: [1] [ENTER].
5. Execute the SEQ command and then eliminate any round-off errors: PROC [NXT] SEQ [1] [⬅] [« »] [0] [MTH] REAL [NXT] RND [ENTER] [PRG] LIST PROC DOLIS.

Result: { 28657 46368 75025 121393 196418 317811 514229 832040 1346269 }

Plotting Sequences

One very good way to gain some insight into a sequence is to plot it.

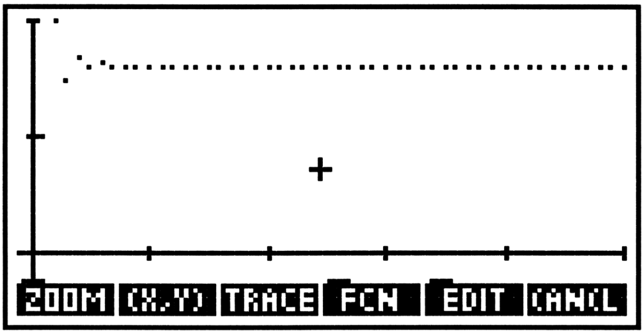
Plotting a sequence with a closed-form definition is very similar to plotting its defining function. In fact, the only difference is that the sequence consists of discrete points, whereas the function is continuous. Thus, when plotting sequences, make sure that CONNECT mode (in the PLOT OPTIONS screen) is *off* (i.e. unchecked).

Example: Plot the sequence formed as the ratio of two consecutive terms of the Fibonacci sequence: F_{n+1}/F_n. This works out as:

$$a_n = \frac{\left(\frac{1+\sqrt{5}}{2}\right)^{n+1} - \left(\frac{1-\sqrt{5}}{2}\right)^{n+1}}{\left(\frac{1+\sqrt{5}}{2}\right)^{n} - \left(\frac{1-\sqrt{5}}{2}\right)^{n}}$$

1. Open the PLOT application and make sure that the TYPE: field is set to Function: [→][PLOT][▲][α][F].
2. Next, highlight the EQ: field and enter the defining function: [▼][←] [EQUATION][▲][←][()][▲][1][+][√x][5][▶][▶][2][▶][▶][y^x][α][←][N][+][1] [▶][−][←][()][▲][1][−][√x][5][▶][▶][2][▶][▶][y^x][α][←][N][+][1][▶][▶][←] [()][▲][1][+][√x][5][▶][▶][2][▶][▶][y^x][α][←][N][▶][−][←][()][▲][1][−][√x] [5][▶][▶][2][▶][▶][y^x][α][←][N][ENTER].
3. Change the INDEP variable to the index variable for the sequence, n: [▼][α][←][N][ENTER].
4. Change H-VIEW to -1 50. The maximum horizontal coordinate is chosen to match the maximum index value you want to plot; 50 seems a good place to start for this sequence. The -1 minimum coordinate allows you to see the *y*-axis.
5. Change V-VIEW to -.5 2. You can estimate the required vertical range by evaluating the defining function at the beginning and ending coordinates (n = 1 and n = 50 in this case). The -.5 minimum coordinate allows you to see the *x*-axis.

6. Move to the PLOT OPTIONS screen and set the plotting range to match the sequence beginning and ending points (-1 and 50). Then make sure that the CONNECT mode is unchecked, and set the STEP interval to 1: OPTS ▶ 1 +/– ENTER 5 0 ENTER ▶ (use ✓CHK, if necessary), then ▼ 1 ENTER.
7. *Optional.* Change the tick-marks to show every 10 units along the horizontal axis and every 1 unit along the vertical axis: ▼ ◀ 1 0 ENTER 1 ENTER (then ✓CHK to uncheck PIXELS, if necessary).
8. Draw the plot: OK ERASE DRAW.



The plot shows clearly that the sequence is nearly constant as the value of the index variable increases. Indeed, the value of terms in the sequence approaches the golden ratio as you increase the index:

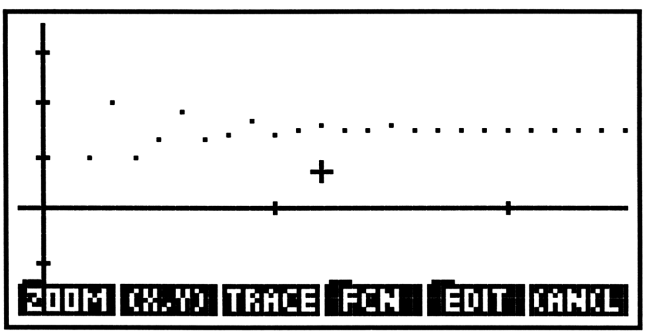
$$\frac{1+\sqrt{5}}{2} \approx 1.61803398875$$

Plotting a recursively-defined sequence is trickier, because it involves plotting a program. The program PSEQ (see page 316), however, simplifies the task. It takes a symbolic object representing the recursive defining function from level 4, a list of the required initial values for the sequence from level 3, the **LO:** value of the plotting range from level 2, and the **HI:** value of the plotting range from level 1. (Note that the objects taken from levels 3 and 4 by PSEQ are the same as those used by →SEQ on levels 3 and 2—recall page 12).

Example: Plot the first 25 terms of the following recursive sequence:

$$
\begin{aligned}
a_1 &= 0 \\
a_2 &= 1 \\
a_3 &= 2 \\
a_n &= \frac{a_1 + a_2 + a_3}{a_2 + a_3}
\end{aligned}
$$

1. First, return to the stack and enter the symbolic form of the recursive defining function: [CANCEL][CANCEL][⬅][EQUATION][▲][α][⬅][A][1][+][α][⬅][A][2][+][α][⬅][A][3][▶][α][⬅][A][2][+][α][⬅][A][3][ENTER].
2. Enter the list of initial values: [⬅][{ }][0][SPC][1][SPC][2][ENTER].
3. Enter the plotting range: [1][ENTER][2][5][ENTER].
4. Plot the sequence using PSEQ: [α][α][P][S][E][Q][ENTER] or [VAR] (then [NXT] or [⬅][PREV] as needed) **PSEQ**.

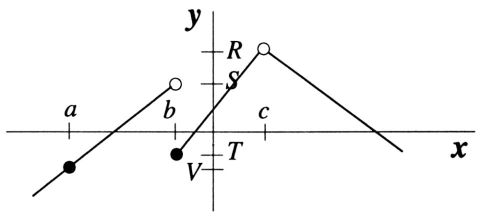


Limits

When looking at a sequence, you will often need to know ask: Are its terms getting ever larger (or smaller) as the index increases, or are they getting ever closer to some fixed value, or *limit*? If such a limit exists for a sequence, then it is useful to help compute the "ultimate" value of a sequence.

But sequences are essentially *functions* of the index variable, and it is an easy jump to see that the concept of the limit can be applied to general functions as well —in the absence of any context of a sequence. Indeed, limits are fundamental to the consideration of one of the most important geometric concepts in calculus: the *instant*—the infinitely precise point.

The idea that a limit is a *prediction* based upon a pattern of behavior is crucial when looking at the limit of a function. Consider the function plot below:



If you were to predict the value of the function at $x = a$ based upon the x-values more negative than a (i.e. approaching a from the left), you would conclude that the value would be V. If you were to predict the value of the function at $x = a$ based upon x-values more positive than a (i.e. approaching a from the right), you would draw the same conclusion. When both predictions—from the left and the right—agree, then you can say that "the limit of the function as x approaches a is V." Thus, in the case of $x = a$, where the function is defined—you may simply substitute a into the function and compute that V is the value, without referring to limits at all.

But what about at points where the function is undefined? Well, if you were to predict the value of the function at $x = c$, approaching both from the left and the right, you would still conclude in both cases that the value of the function at $x = c$ is R. Thus R is the limit of the function as x approaches c. However, if you were to predict the value of the function at $x = b$, approaching from the left, you'd say the value is S. If you were to predict the value from the right you'd say T. In this case, because the two predictions don't agree, the function has no limit at $x = b$. This is true even though the function has the value T at $x = b$.

In mathematical notation, the limit of a function, $f(x)$, at a given point, a, is:

$$\lim_{x \to a} f(x)$$

To express the limit of the function, approached from only one side (left or right), a plus (right) or minus (left) is added to the notation:

$$\lim_{x \to a^-} f(x) \qquad \lim_{x \to a^+} f(x)$$

Continuity and Limits

A function, such as the one just illustrated, is *continuous at a point* when the value of the function at that point equals the value of the limit at that point. That is, the following three requirements must be met for a function $f(x)$ to be continuous at a point a:

1. The limit of $f(x)$ as x approaches a must exist.
2. $f(a)$ must be defined.
3. The limit at a must equal $f(a)$.

Thus the function illustrated in the previous plot is:

- *continuous* at a, because the limit at a (V) equals its value at a (V).
- *discontinuous* at b, because although defined at b (T), it has no limit at b.
- *discontinuous* at c, because although it has a limit at c (R), it is undefined at c. Note that this kind of discontinuity can be *removed* if you can add to the function definition a value for c.

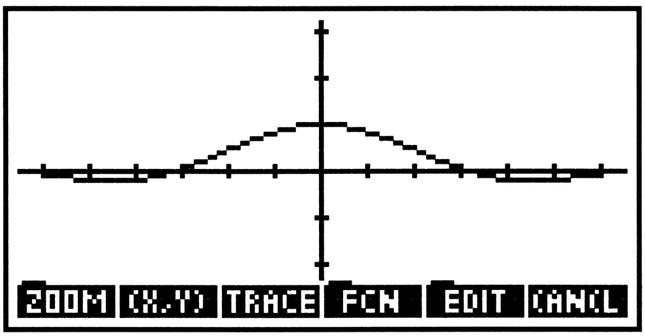
Just as limits can be separated into single-sided forms ("limit from the left" or "limit from the right"), so can continuity at a point. For example, in the plotted function shown earlier, the function is continuous at b from the *right* because the limit from the right (T) does match the value of the function at b. It is still discontinuous from the left, however.

Finding Limits Graphically

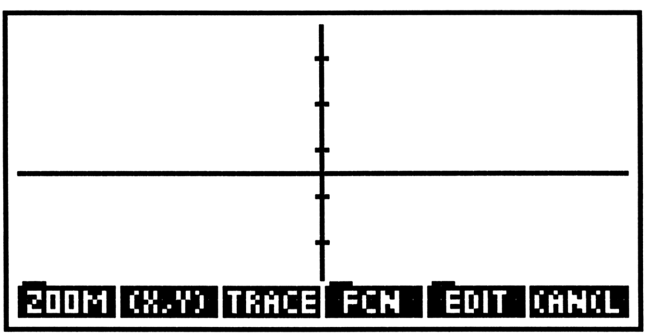
Of course, to use the HP 48 to find limits visually (graphically), you use the zooming features to magnify the area around the limit point of a function.

Example: Find the following limit graphically: $\lim_{x \to 0} \frac{\sin x}{x}$

1. Open the PLOT application, highlight the TYPE: field and set it to Function. Then reset the plot parameters to their defaults: CANCEL ↱ PLOT ▲ α F DEL ▼ ENTER.
2. At EQ:, enter the function: ▼ ' SIN α ↰ X ▶ ÷ α ↰ X ENTER.
3. Change the INDEP variable to x (lower-case): α ↰ X ENTER.
4. In radians mode (↰ RAD, if needed), plot: ERASE DRAW.



5. Use Box-Zoom to draw a small box around the part of the function near $x = 0$: Move the cursor (via ▲ and ◀) to a point just above and left of the point where the function appears to cross the y-axis. Press ZOOM BOXZ, then draw the box via ▼ and ▶, and press ZOOM.



6. The region around the point $x = 0$ appears to be flat at high magnification. Find the value of the function at this point with the TRACE feature: [▶] TRACE (X,Y). Result: X: 1.54E-3   Y: 1.00E0

   The y-coordinate is 1.00 for the line shown, so the limit for the function when x approaches zero appears to be 1.

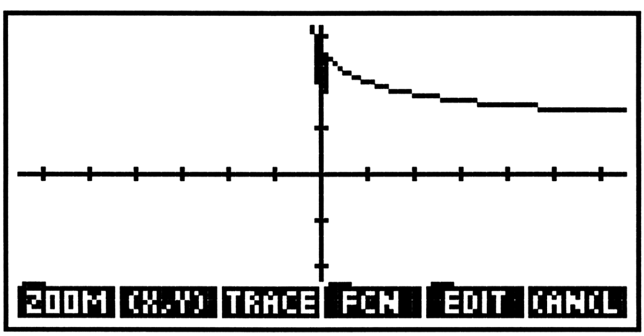
7. *Optional.* Note what the display says if you place the cursor directly on $x = 0$: Press [◀].   Result: X: 0   Y:

   The function is undefined at $x = 0$ (although you can't see the "hole" in the plot, because it coincides with the axis).

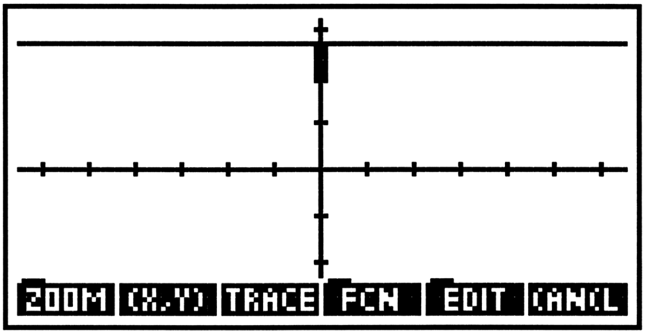
Finding limits graphically in this way works well for many functions. But there is one important caveat: Beware the limitations of the machine's precision! Twelve digits is a lot of precision, but it is nowhere near infinity. Some functions are more subject to rounding error irregularities around the 12th digit than are others—irregularities that make for peculiar plots during zooming.

Example: Find the following limit graphically: $\lim_{x\to 0}(1+x)^{\frac{1}{x}}$.

1. Return to the PLOT screen and rest the plot parameters: [CANCEL] [DEL] [▼] [ENTER].
2. Highlight the EQ: field and enter the function: [▲] ['] [←] [()] [1] [+] [α] [←] [X] [▶] [y^x] [←] [()] [1] [÷] [α] [←] [X] [ENTER].
3. Enter the INDEP variable, be sure the CONNECT option is checked, and plot: [α] [←] [X] [ENTER] OPTS [▼] ✓CHK [ENTER] ERASE DRAW.

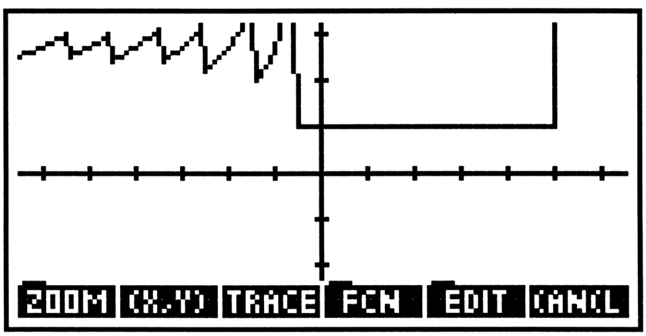


4. To keep track of the amount of zooming this time, use the ZFACTor: Press **ZOOM** **ZFACT** 1 0 0 0 0 0 0 ENTER ENTER to change the horizontal zoom factor to 10^6, so that for each horizontal zoom-in (HZIN), the horizontal scale will shrink by a factor of 1,000,000.
5. Now zoom-in horizontally by a factor of 1,000,000: NXT **HZIN**.



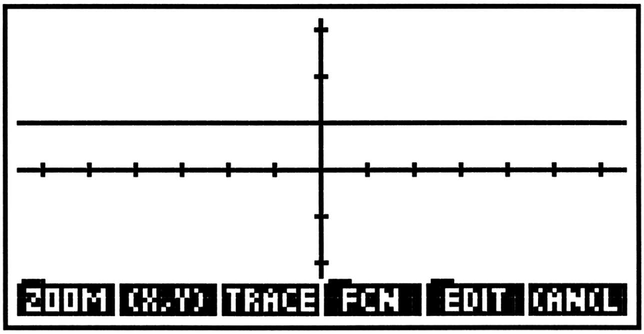
6. The graph appears to be constant except near $x = 0$. Use TRACE and (X,Y) to determine the value of the constant value: ▶ ▶ ▶ ▶ **TRACE** **(X,Y)**. Result: X: .0000004   Y: 2.7182812848

   The apparent value of the limit here seems to be very close to e, the natural logarithm base, whose value, to 12 digits, is 2.71828182846.
7. However, suppose that you zoom in much closer to zero. Repeat the horizontal zoom-in: NXT **ZOOM** NXT **HZIN**.



What you see now is round-off error. Because you have zoomed by a total factor of 10^{12}, the magnitude of the x's being plotted is near the maximum 12-digit precision of the HP 48. The sawtooth appearance reflects the rounding effect, not the behavior of the function.

8. Notice also that as the plot crosses the y-axis, it seems constant, a sign that it may be productive to zoom one more time. Perform one more horizontal zoom-in: ZOOM (NXT) HZIN.



9. Determine its value using the TRACE feature: (▶) TRACE (X,Y).

   Result: X: 1.E-16 Y:1

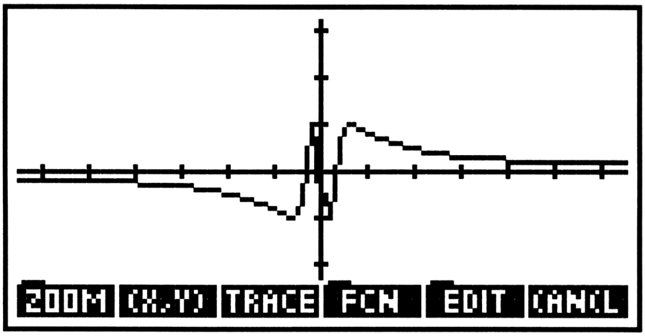
   If you were to happen upon this constant graph, you would conclude that the limit of the function is 1.

So which is the correct limit of this function: e or 1? Trust the limit computed without the effects of round-off error: e. You'll see more of this function later.

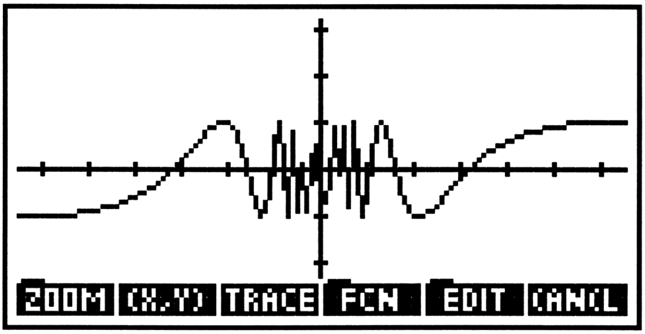
Although the previous example showed a particularly tricky function for determining a limit, there are some functions for which zooming doesn't work at all—because they have no predictable pattern when approaching the limit point (i.e. they have no limit). You'll recognize these functions as you begin zooming in on the limit point.

Example: Find the following limit graphically: $\lim_{x \to 0} \sin\left(\frac{1}{x}\right)$.

1. Return to the PLOT screen and reset the plot parameters: (CANCEL) (DEL)(▼)(ENTER).
2. Enter the function into the EQ: field and enter the correct INDEP variable: (')(SIN)(1)(÷)(α)(←)(X)(ENTER)(α)(←)(X)(ENTER).
3. Plot the function: ERASE DRAW.

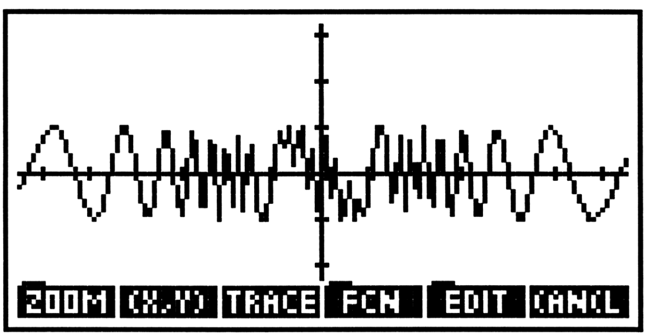


4. Change the horizontal zoom factor to 10 and magnify the area around the y-axis: ZOOM ZFACT 1 0 ENTER ENTER NXT HZIN.



Instead of smoothing the curve, zooming has the effect of magnifying the oscillating behavior in the neighborhood of the limit point.

5. Zoom in horizontally again to confirm this: ZOOM NXT HZIN.



Yes, the situation is deteriorating—this function probably has no limit. After all, what kind of reliable prediction can you make when the pattern gets increasingly wild as you approach the limit point?

Finding Limits Numerically

Plotting functions to graphically examine their behaviors around a candidate limit point can be a useful for identifying potential troublemakers—if you are careful not to introduce round-off error. Finding limits numerically on the HP 48 presents similar problems—round-off effect and oscillating behavior.

This book represents a limit expression on the HP 48 as a two-element list. For example, the expression, $\lim_{x \to a} f(x)$, is represented as `{ 'f(x)' 'x=a' }`. To indicate a limit of positive or negative infinity, use the `'i'` constant. Thus you might use `'x=i'`, `'n=-i'`, etc., as the second element in a limit expression list.

The program `LIM` (see page 300) computes the limit of a function and displays the "moving" results of approaching the limit point. `LIM` takes the list representing the limit expression (see above) from level 3, a starting magnitude for the search on level 2, and an ending magnitude for the search on level 1. These magnitudes tell the HP 48 the range of values of the function variable to use in determining the function's limit behavior at the given point. The values you input are the "orders of magnitude" for how close you wish to approach the limit point.

For example, if the limit point is infinity, a range of 1 to 11 would have the HP 48 search values of the function variable from roughly 10^1 to 10^{11}—spending equal time with each intervening order of magnitude. Remember that above 10^{12}, round-off error begins to affect the computations for all functions (and many functions are affected by round-off error at much smaller orders of magnitude). For limits that approach a finite number, a range of 1 to 11 would have the HP 48 search values of the function variable from within 10^{-1} to 10^{-11} of the finite number.

To compute the left-hand limit, use a negative number as the starting value; for the right-hand limit, use a positive number. In both cases, the absolute values of the starting and ending values are used to set the search range.

The program displays the search progress in the top part of the screen, then returns a list of approximations for the limit. These values are representative and are spread evenly throughout the search range, unless the search found a value for the limit which didn't change anymore, in which case the repetitions aren't included.

The examples that follow illustrate not only how to find limits numerically using LIM, but also how to recognize when round-off is clouding the picture, when the function has no limit, and when to change the region of the search.

Example: Use LIM to compute $\lim_{x \to 0}(1+x)^{\frac{1}{x}}$. Compare the results using search ranges of 1-3, and 6-16.

1. Enter the limit list and make a copy: ⬅ { } ' ⬅ ( ) 1 + α ⬅ X ▶ y^x ⬅ ( ) 1 ÷ α ⬅ X ▶ ▶ ' α ⬅ X ⬅ = 0 ENTER ENTER.
2. Enter the search range (orders of magnitude): 1 ENTER 3 ENTER.
3. Search for the limit: α α L I M ENTER or VAR (NXT or ⬅ PREV as needed) LIM.

   Result: { 2.5937424601 2.63720428005
   2.66607579515 2.68490944975
   2.69704918383 2.700481382942
   2.70975552942 2.71289066516
   2.71487566137 2.71613085585
   2.71692393224 }

   The approximations are beginning to move towards a value slightly less than 2.72, but perhaps you should expand the search magnitude.
4. Drop the previous result, enter the new search range, and begin the expanded search: ⬅ 6 ENTER 1 6 ENTER LIM.

   Result: { 2.71828046932 2.71828169254
   2.71828181487 2.7182818271
   2.71828182832 2.71828182845 1 }

   The approximation gets much better as the search goes from 10^{-6} to 10^{-11}, but then what happens? Just as the graphical approach encounters round-off error too close to the limit point, so LIM suffers if you search magnitudes beyond the machine's precision. The maximum magnitude is 11; for many functions, the *trustworthy* magnitude is less. (But if you use a range of 1 - 11 to begin with, it gives you a good overall picture of the function at the given point. Usually you can tell where the round-off error begins in the returned list.)
   Conclusion: Limit ≈ 2.71828182845 (note: $e \approx 2.71828182846$).

Example: Use LIM to compute $\lim_{x \to 1} \frac{1 - 1/x^2}{1 - x}$.

1. Enter the limit expression list: ⬅ { } ' ⬅ ( ) 1 − 1 ÷ α ⬅ X y^x 2 ▶ ÷ ⬅ ( ) 1 − α ⬅ X ▶ ▶ ' α ⬅ X ⬅ = 1 ENTER.
2. Enter the search range: 1 ENTER 1 1 ENTER.
3. Search for the limit: α α L I M ENTER or VAR (then NXT or ⬅ PREV as needed) LIM.

   Result: { -1.73553719008 -1.9703950593 -1.997003995 -1.99970004 -1.99997 -1.999996 -2 }.

   Unlike the previous example, when the search settled on a fixed value, it was an extension of the trend developed by the previous values in the search and not a break from it.

   Conclusion: Limit = -2.

Example: Use LIM to compute $\lim_{x \to 0} \frac{\sin x}{x}$.

1. Enter the limit expression list: ⬅ { } ' SIN α ⬅ X ▶ ÷ α ⬅ X ▶ ' α ⬅ X ⬅ = 0 ENTER.
2. Enter the search range: 1 ENTER 1 1 ENTER.
3. Search for the limit: α α L I M ENTER or VAR (then NXT or ⬅ PREV as needed) LIM.

   Result: { .998334166468 .99998333417 .999999833333 .999999998333 .999999999983 1 }

   Conclusion: The limit is 1.

For some functions you may need to examine the right-hand and left-hand limits to be sure that they match before deciding if the given point has a general limit.

Example: Compute and compare $\lim\limits_{x \to 2^-} \dfrac{x-2}{|x-2|}$ and $\lim\limits_{x \to 2^+} \dfrac{x-2}{|x-2|}$

1. Enter the limit expression list and make an extra copy: [←][{ }]['] [←][( )][α][←][X][−][2][▶][÷][MTH] [VECTR] [ABS] [←][( )][α][←][X][−][2] [▶][▶]['][α][←][X][←][=][2][ENTER][ENTER].
2. Enter the search range using a positive value for the starting point: [1][ENTER][1][1][ENTER].
3. Search for the right-hand limit (+): [VAR] [LIM]. <u>Result</u>: `{ 1 }`.

   The right-hand limit—1—was found very quickly and conclusively.
4. Drop the previous result, enter the search range using a negative value for a starting point, and repeat the search: [⬅][1][+/−][ENTER] [1][1][ENTER] [LIM]. Result: `{ -1 }`.

   The left-hand limit (-1) was found quickly but it doesn't match the right-hand limit. Thus the function has no overall limit at $x = 2$.

Example: Compute the following limit: $\lim\limits_{x \to \infty} \dfrac{(3x-2)^2}{(3-x)(2-x)}$

1. Enter the limit list (note how `'i'` indicates infinity): [←][{ }]['][←][( )] [3][×][α][←][X][−][2][▶][y^x][2][÷][←][( )][←][( )][3][−][α][←][X][▶][×] [←][( )][2][−][α][←][X][▶][▶][▶]['][α][←][X][←][=][α][←][I][ENTER].
2. Enter the search range: [1][ENTER][1][1][ENTER].
3. Search for the limit: [α][α][L][I][M][ENTER] or [VAR] (then [NXT] or [←][PREV] as needed) [LIM].

   <u>Result</u>: `{ 14 9.34188933305 9.0331153782`
   `9.00330115038 9.0003300115`
   `9.00003300011 9.0000033 9.00000033`
   `9.000000033 9.0000000033 9.0000000033`
   `}`

   The limit being approached is clearly 9.

There are several types of situations where a point doesn't have a limit: **(i)** vertical asymptotes; **(ii)** jump discontinuities; **(iii)** holes; and **(iv)** chaotic oscillation. The next few examples show how to recognize these situations with the `LIM` program.

Example: Use `LIM` to compute $\lim_{x\to 1} \dfrac{1+\frac{1}{x}}{1-x^2}$.

1. Enter the limit expression list: [←][{ }][′][←][( )][1][+][1][÷][α][←][X][▶][÷][←][( )][1][−][α][←][X][y^x][2][▶][▶][′][α][←][X][←][=][1][ENTER].
2. Enter the search range: [1][ENTER][1][1][ENTER].
3. Search for the limit: [VAR] ([NXT] or [←][PREV]) **LIM**. Result:

```
{ -9.0909090909 -99.00990099 -999.000999
  -9999.0001 -99999.00001 -999999.5
  -9999999.5 -99999999.5 -999999999.5
  -9999999999.5 -99999999999.5 }
```

Whenever you see the list of values increasing (or decreasing) in magnitude by a factor of 10 with each entry, you can conclude that there is no limit for the function at the given point.

Example: Use `LIM` to compute $\lim_{x\to 0} \sin\left(\dfrac{1}{x}\right)$.

1. Enter the limit expression as a list: [←][{ }][′][SIN][1][÷][α][←][X][▶][′][α][←][X][←][=][0][ENTER].
2. Enter the search range: [1][ENTER][1][1][ENTER].
3. Search for the limit: [α][α][L][I][M][ENTER] or [VAR] ([NXT] or [←][PREV] as needed) **LIM**.

Result:
```
{ -.544021110889 -.50636564111
  .826879540532 -.305614388888
  .035748797972 -.349993502171
  .420547793191 .93163902711
  .545843449449 -.487506025088
  .928693660497 }
```

There is no trend showing in the list at all—just a set of numbers between 1 and -1. Conclusion: There is no limit.

Series

A *series* is the sum of all the terms in a sequence. In a sequence with infinitely many terms, its series is the *limit of the partial sums of the terms in the sequence.*

In mathematical notation, a series is a summation: $\sum_{n=i}^{j} a_n$, where a_n is the nth term of a sequence whose first term is a_i and whose last term is a_j. If j is ± infinity, then the series becomes the limit of a summation: $\lim_{N\to\infty} \sum_{n=1}^{N} a_n$ where the sequence for which you are computing the limit is a sequence of sums:

$$\left\{ \sum_{n=1}^{1} a_n \quad \sum_{n=1}^{2} a_n \quad \sum_{n=1}^{3} a_n \quad \cdots \quad \sum_{n=1}^{N} a_n \quad \cdots \right\}$$

One of the most important qualities of a series is whether it *converges* to a particular, finite, value, or whether it *diverges* to ±infinity. Or, to put it another way, does the *limit* of the *sums* exist?

Of course, you can use a program similar to LIM to compute a running total as the index increases and watch its progress to see if it converges (indeed, such a program is presented below). But there is no efficient way to compute all sums; some converge very slowly and there is no way to "jump" to the end to find the result.

Moreover, you cannot assume that because the terms of a sequence converge to a limit (i.e. they stop getting larger) that the *sum of the terms* converges to a limit (i.e. stops getting larger). In fact, in order for a series to *even possibly* converge, its associated sequence must converge to zero. But just because a sequence converges to zero, doesn't mean it does so "fast" enough for the associated series to converge.

For example, consider $\sum_{n=1}^{\infty} \frac{1}{n^6}$. After summing the first 78 terms, the total stops changing within the precision of the HP 48; it *appears* to converge.

You would expect $\sum_{n=1}^{\infty} \frac{1}{n^5}$ to converge—if at all—more slowly than $\sum_{n=1}^{\infty} \frac{1}{n^6}$.

Indeed it does. This table illustrates the *apparent* convergent values for the series $\sum_{n=1}^{\infty} \frac{1}{n^p}$ where $p =$ 1, 2, 3, 4, 5, 6, and 7. (Remember: the terms being added approach zero—so convergence is possible but not guaranteed.)

Series	HP 48 Computed Limit	# of Terms Used
$\sum_{n=1}^{\infty} \frac{1}{n}$	24.5322108226	20,000,000,000
$\sum_{n=1}^{\infty} \frac{1}{n^2}$	1.64493215409	447,215
$\sum_{n=1}^{\infty} \frac{1}{n^3}$	1.20205689144	5,850
$\sum_{n=1}^{\infty} \frac{1}{n^4}$	1.08232323371	670
$\sum_{n=1}^{\infty} \frac{1}{n^5}$	1.03692775496	184
$\sum_{n=1}^{\infty} \frac{1}{n^6}$	1.01734306194	78
$\sum_{n=1}^{\infty} \frac{1}{n^7}$	1.00834927740	43

The HP 48 shows that all of these series converge, but do they really—or is the appearance a side-effect of the limited precision of the machine?

The first series, $\sum_{n=1}^{\infty} \frac{1}{n}$ (the *harmonic series*), actually diverges. The terms in its sequence simply don't shrink fast enough for the sum to converge. The moral here: You can't completely trust a machine to tell you whether a series converges.

However, there are a group of tests you can apply to a series to assist in determining whether it converges. If you can conclude that it does, then the machine is useful in finding an approximation for the series. The tests, which you will find discussed and derived in any first-year calculus text, are summarized below:

Nth Term test: If the limit of the terms of the sequence $\neq$ zero, the series diverges.

Root test: If $\lim_{n\to\infty} \sqrt[n]{|a_n|}$ is less than one, the series converges; if greater than one, the series diverges; if equal to one, the test is inconclusive.

Ratio test: If $\lim_{n\to\infty} \left| \frac{a_{n+1}}{a_n} \right|$ is less than one, the series converges; if greater than one, the series diverges; if equal to one, the test is inconclusive.

Comparison: If you can find a convergent series whose terms are all greater than or equal to the corresponding terms in the series you're testing, then the test series converges. If you can find a divergent series whose terms are all less than or equal to the corresponding terms in the test series, then the test series diverges.

A few additional tests are valid if the series you're testing meets certain criteria:

Limit Comparison: Every term, a_n, of the test series must be greater than zero. Find a different series all of whose terms, b_n, are greater than zero and compute the limit, L, of a_n/b_n. If $0 \leq L < \infty$ and the comparison series converges, then the test series converges. If $0 < L$ and the comparison series diverges, then the test series diverges.

Integral: The defining function, $f(x)$, for the series must be continuous, positive and decreasing for this test to be valid. If the improper integral $\int_1^{\infty} f(x)\,dx$ converges, the series also converges. If it diverges, so does the series.

While the HP 48 can't directly prove that a given series converges or diverges, it can help you indirectly—by performing some of the aforementioned tests. Three programs, SERX1, SERX2, and SERX3, are provided to help determine whether or not a series converges.

The program SERX1 (page 323) performs the Root test on the given series. It takes the symbolic defining function for the series from level 2, and the index variable from level 1 and returns one of the following messages:

"Diverges", "Converges", or "Inconclusive".

SERX2 (page 324) performs the Limit Comparison test given two series with positive terms. It takes the symbolic defining function of the series being tested from level 3, the symbolic defining function of the comparison series from level 2, and the index variable (which should be the same for both series) from level 1 and computes the limit of the quotient of these series. The list returned to level 1 contains representative values of the limit as the index uses higher and higher values —similar to the list LIM returns (page 24). You can then judge convergence or divergence on the basis of the trend you see in the list. Note that SERX2 makes no attempt to confirm that the series you are using are valid for use with the Limit Comparison test.

SERX3 (page 324) performs the Integral test on the given series (as long as it is continuous, positive and decreasing). It takes the symbolic defining function of the series from level 2 and the index variable from level 1 and computes the value of the definite integral using a selected set of increasing intervals. It returns a list containing representative values as the interval is increased—analogous to those returned by LIM and SERX2. Note that SERX3 may take awhile for certain functions. If an unusually large delay occurs, you may be better off solving the integral by hand if you can and using LIM to test for convergence or divergence.

When using these programs to determine whether a series converges or not, it is best to use them in "numerical" order, because of the speed with which each operates. The root test (SERX1) is fastest, and the integral test (SERX3) is usually the slowest.

Example: Does the series $\sum_{n=1}^{\infty} \frac{2^{3n}}{7^n}$ converge or diverge?

1. Enter the defining function: [←][EQUATION][2][y^x][3][α][←][N][▶][÷][7][y^x][α][←][N][ENTER].
2. Enter the index variable: ['][α][←][N][ENTER].
3. Perform the Root test (SERX1): [α][α][S][E][R][X][1][ENTER] or [VAR] (then [NXT] or [←][PREV] as needed) [SERX1].

Result: "Diverges"

Example: Does the series $\sum_{n=1}^{\infty} \frac{1}{n}$ converge or diverge?

1. Enter the defining function and the index variable: ['][1][÷][α][←][N][ENTER]['][α][←][N][ENTER].
2. Perform the Root test (SERX1): [VAR] [SERX1].

Result: "Inconclusive"

Because the Root test in inconclusive, your next choice would be the Limit Comparison test (SERX2) *if you can find an appropriate series to use as a comparison*. In this particular case, a good comparison series does not readily leap to mind. Thus the next option is to use the Integral test (SERX3).

3. Enter the defining function and index variable again: ['][1][÷][α][←][N][ENTER]['][α][←][N][ENTER].
4. Perform the Integral test: [α][α][S][E][R][X][3][ENTER] or [VAR] (then [NXT] or [←][PREV] as needed) [SERX3].

Result: { 2.3 46.1 184.2 460.5 }

Because there is no sign of convergence in the list, you can conclude that the series diverges.

Example: Does the series $\sum_{n=1}^{\infty} \frac{1}{n^2}$ converge or diverge?

1. Enter the defining function and the index variable: ['] [1] [÷] [α] [←] [N] [yˣ] [2] [ENTER] ['] [α] [←] [N] [ENTER].
2. Make an extra copies of the defining function and index variable (just in case the Root test is inconclusive): [←] [STACK] [NXT] DUP2.
3. Perform the Root test (SERX1): [VAR] SERX1.

   Result: "Inconclusive"

   Because the Root test in inconclusive, use the Limit Comparison test. Since you just demonstrated that $\sum_{n=1}^{\infty} \frac{1}{n}$ diverges, perhaps it would make a good comparison series.
4. Drop the previous result, enter the comparison function, swap it to level 2, and perform the Limit Comparison test (SERX2): [←] ['] [1] [÷] [α] [←] [N] [ENTER] [SWAP] SERX2.

   Result: { .1 .01 .001 .0001 .00001 .000001 .0000001 .00000001 .000000001 .0000000001 }

   Clearly, the limit is zero—which means the test is inconclusive. Because the comparison series is a divergent one, a positive limit would have meant the test series is divergent, but a zero limit means that the test is inconclusive. So, on to the Integral test.
5. Enter the function and its index again: ['] [1] [÷] [α] [←] [N] [yˣ] [2] [ENTER] ['] [α] [←] [N] [ENTER].
6. Perform the Integral test (SERX3): [VAR] SERX3.

   Result: { 0.9 1.0 1.0 1.0 }

   The series converges according to the Integral test.

Example: Does the series $\sum_{n=1}^{\infty} \frac{n}{2n^2 - 1}$ converge or diverge?

1. Enter the defining function and the index variable and make a set of copies: ['] [α] [←] [N] [÷] [←] [( )] [2] [×] [α] [←] [N] [y^x] [2] [−] [1] [ENTER] ['] [α] [←] [N] [ENTER] [←] [STACK] [NXT] **DUP2**.
2. Perform the Root test (SERX1): [VAR] **SERX1**.

   Result: "Inconclusive"

3. Drop the previous result and enter a function with which to compare limits ($1/n$ seems a good choice): [◄] ['] [1] [÷] [α] [←] [N] [ENTER] [SWAP].
4. Perform the Limit Comparison test (SERX2): [VAR] **SERX2**.

   Result (to 3 places): { .503 .500 .500 .500 .500 .500 .500 }

   Since the Limit Comparison test returns a positive limit (0.5) with a comparison series that diverges, the test series must also diverge.

Example: Does the series $\sum_{n=1}^{\infty} \frac{1}{4n^2 + 9}$ converge or diverge?

1. Enter the defining function and the index variable and make an extra set of copies: ['] [1] [÷] [←] [( )] [4] [×] [α] [←] [N] [y^x] [2] [+] [9] [ENTER] ['] [α] [←] [N] [ENTER] [←] [STACK] [NXT] **DUP2**.
2. Perform the Root test (SERX1): [VAR] **SERX1**.

   Result: "Inconclusive"

3. Drop the previous result and enter a series with which to compare limits ($1/n^2$ looks to be a good choice here): [◄] ['] [1] [÷] [α] [←] [N] [y^x] [2] [ENTER] [SWAP].
4. Perform the Limit Comparison test (SERX2): [VAR] **SERX2**.

   Result (to 3 places): { .244 .250 .250 .250 .250 .250 .250 0.250 }

   Because the limit is positive and finite (0.25) and the comparison series converges, then the test series converges.

Computing Series

Once you have assured yourself that a particular series converges, the HP 48 can help you approximate the infinite sum.

The program `SERIES` (see page 323) takes the defining function for a series from level 4, the index variable from level 3, the beginning index value from level 2, and the number of decimal places to which you want to approximate the series from level 1. It returns the approximate value of the series to level 2 and the final index value used to compute the approximation to level 1. The larger the number of decimal places given in level 1 (i.e. the more precision you require) the longer it will take to compute the approximation.

Look at some examples . . .

Example: Approximate the following infinite sum to three decimal places:

$$\sum_{k=1}^{\infty}\left(\frac{k}{2k+1}\right)^{k}$$

1. Enter the defining function: [⬅] [EQUATION] [⬅] [( )] [α] [⬅] [K] [÷] [2] [α] [⬅] [K] [+] [1] [▶] [▶] [y^x] [α] [⬅] [K] [ENTER].
2. Enter the index variable: ['] [α] [⬅] [K] [ENTER].
3. Enter the beginning index value: [1] [ENTER].
4. Enter the number of decimal places you want in the approximation: [3] [ENTER].
5. Execute `SERIES`: [α] [α] [S] [E] [R] [I] [E] [S] [ENTER] or [VAR] (then [NXT] or [⬅] [PREV] as needed) SERIE.

   Result:
   ```
   2:          0.650
   1:         22.000
   ```
   The approximate value of the infinite sum is `0.650` which was arrived at after summing the first 22 terms.

Example: Approximate the following infinite sum to two decimal places:

$$\sum_{k=1}^{\infty} \frac{2k+\sqrt{k}}{k^3+\sqrt{k}}$$

1. Enter the defining function: [←][EQUATION][▲][2][α][←][K][+][√x][α][←][K][▶][▶][α][←][K][y^x][3][▶][+][√x][α][←][K][ENTER].
2. Enter the index variable: ['][α][←][K][ENTER].
3. Enter the beginning index value: [1][ENTER].
4. Enter the number of decimal places you want in the approximation: [2][ENTER].
5. Execute SERIES: [VAR] [SERIE].

Result:
```
2:        3.00
1:      205.00
```

The infinite sum, approximated to two decimal places is 3.00

Example: Approximate the following infinite sum to five decimal places:

$$\sum_{k=1}^{\infty} \frac{2^k k!}{k^k}$$

1. Enter the defining function: ['][2][y^x][α][←][K][×][α][←][K][MTH][NXT][PROB][!][÷][α][←][K][y^x][α][←][K][ENTER].
2. Enter the index variable: ['][α][←][K][ENTER].
3. Enter the beginning index value: [1][ENTER].
4. Enter the number of decimal places you want in the approximation: [5][ENTER].
5. Execute SERIES: [VAR] [SERIE].

Result:
```
2:   12.94895
1:   58.00000
```

Using Series to Approximate Functions

The easiest functions for calculators to compute and manipulate are polynomials. They're composed only of additions and multiplications. But non-polynomial functions such as trigonometric, exponential, and logarithmic functions are a different matter. In fact, most calculator routines for computing non-polynomial functions actually use *polynomial approximations* of those functions.

Example: Find a polynomial approximation for the function $f(x) = \sin x$.

Result: $$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots = \sum_{k=0}^{\infty} (-1)^k \frac{x^{2k+1}}{(2k+1)!}$$

Example: Find a polynomial approximation for the function $f(x) = e^x$.

Result: $$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \cdots = \sum_{k=0}^{\infty} \frac{x^k}{k!}$$

Example: Find a polynomial approximation for the function $f(x) = \ln(1 + x)$.

Result: $$\ln(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \cdots = \sum_{k=0}^{\infty} (-1)^k \frac{x^{k+1}}{k+1}$$

Each of these polynomial approximations is a *power series*—a polynomial with an infinite number of terms. More precisely, a power series is the limit of a sequence of polynomials: $\sum_{n=0}^{\infty} c_n x^n = \lim_{n\to\infty} p_n(x)$, where $p_n(x)$ is a polynomial of degree n, and c_n is a coefficient of a term in the polynomial. To obtain an approximation using a power series at a particular real value, x_0, simply substitute x_0 into the power series for x and find the limit of the series to whatever number of decimal places you desire.

Example: Approximate the value of sin(2) to four decimal places, using the power series shown in the example above.

1. Enter the defining function for the power series that approximates the sine function, remembering to substitute 2 for x in the definition: [←][EQUATION][←][( )][+/−][1][▶][y^x][α][←][K] [▶][▲][2][y^x][2][α][←][K] [+][1][▶][▶][←][( )][2][α][←][K][+][1][▶][α][←][DEL][ENTER].
2. Enter the index variable for the series: ['][α][←][K][ENTER].
3. Enter the beginning index value and the desired number of decimal places in the approximation: [0][ENTER][4][ENTER].
4. Execute SERIES to compute the approximation: [VAR] [SERIE].

   Result: 2: 0.9093
   1: 8.0000

   After 8 terms, the approximation of sin(2) is 0.9093.
5. Now check the approximation using the SIN function on the HP 48. Make sure that your in Rad mode first: ([←][RAD], if necessary) [2] [SIN]. Result: 0.9093 — a match!

Example: Find ln(3) to four places, using the power series for ln(1+x) as described above (i.e. replace x in the defining function with 2).

1. Enter the defining function for the power series that approximates the ln(1+x) function, remembering to substitute 2 for x in the definition: [←][EQUATION][←][( )][+/−][1][▶][y^x][α][←][K] [▶][▲][2][y^x][α][←] [K][+][1][▶][▶][α][←][K][+][1] [ENTER].
2. Enter the index variable for the series: ['][α][←][K][ENTER].
3. Enter the beginning index value and the desired number of decimal places in the approximation: [0][ENTER][4][ENTER].
4. Execute SERIES to compute the approximation: [VAR] [SERIE].

   "Wait a second! What's happening?"

   You are working with a divergent series, and SERIES assumes that the series you're using is convergent. Press [ENTER] (or any key other than [CANCEL]) to end the search, then [◄][◄][◄] to clean up.

This example illustrates an important aspect about power series—they must converge for the value of x you're using in order to work as an approximation. Some power series are only *valid* as approximations for values within a particular interval. To see this, repeat the previous example using $x = 0.5$ instead of $x = 2$.

Example: Approximate ln(1.5) to four decimal places using the power series for ln(1+x) described above. That is, replace x in the defining function with 0.5.

1. Enter the defining function for the power series that approximates the ln (1+x) function, remembering to substitute 0.5 for x in the definition: [←] [EQUATION] [←] [()] [+/−] [1] [▶] [y^x] [α] [←] [K] [▶] [▲] [•] [5] [y^x] [α] [←] [K] [+] [1] [▶] [▶] [α] [←] [K] [+] [1] [ENTER].
2. Enter the index variable for the series: ['] [α] [←] [K] [ENTER].
3. Enter the beginning index value and the desired number of decimal places in the approximation: [0] [ENTER] [4] [ENTER].
4. Execute SERIES to compute the approximation: [VAR] [SERIE].

   Result: 2: 0.4055
   1: 20.0000

   This time the power series converges after 20 terms.
5. Check the approximation using the LN function on the HP 48: [1] [•] [5] [→] [LN]. Result: 0.4055 — a match!

How does one find out the *interval of validity* for a given approximation?

One reasonably good method is to plot the original function and a partial sum of the power series (i.e. the first few terms of the series) together and visually note the interval where they coincide.

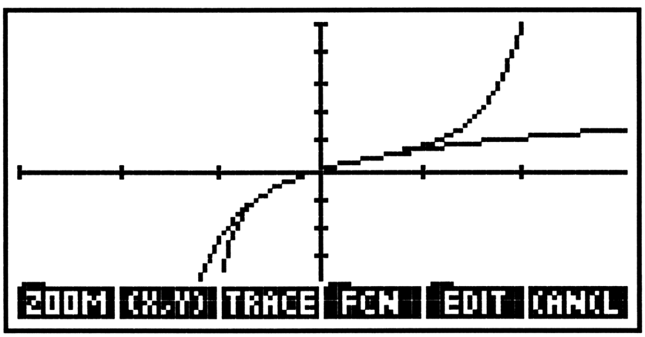
For example, to find the interval of validity of the power series for ln (1 + x), you would plot the function, ln $(1 + x)$, and the polynomial representing the partial sum of the power series that includes the first, say, five terms: $x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \frac{x^5}{5}$. The next example illustrates what you can find. . .

Example: Plot the function, $\ln(1+x)$, and the first five terms of its power series,

$$x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \frac{x^5}{5}$$

to visually determine the interval of validity.

1. Open the PLOT application and make sure that the **TYPE:** field is set to Function: [→][PLOT][▲][α][F].
2. Highlight the **EQ:** field and enter a list containing both functions: [▼] [←][{ }]['][→][LN][1][+][α][←][X][▶][▶]['][α][←][X][−][α][←][X][y^x][2][÷] [2][+][α][←][X][y^x][3][÷][3][−][α][←][X][y^x][4][÷][4][+][α][←][X][y^x][5] [÷][5][ENTER].
3. Set **INDEP:** to x (lower-case), **H-VIEW** to −3 3, and **V-VIEW** to −5 5. Move the **PLOT OPTIONS** screen and set **H-TICK:** to 1, **V-TICK:** to 1, and uncheck the **PIXELS** field on the last line. This will display tick-marks every unit (instead of every 10 pixels) on both the horizontal and vertical axes.
4. Press **OK** **ERASE** **DRAW** to draw the plot.



Note that the two graphs coincide roughly between $x = -1$ and $x = 1$, which indicates that these roughly represent the boundaries for the interval of validity. Outside of these boundaries, the power series diverges and is useless as an approximation.

Taylor Series

Although the examples so far have used well-known functions to illustrate the use of power series to approximate function values, the technique is more commonly used to approximate otherwise unknown functions.

The power series you have seen so far are referred to collectively as *Taylor series*. The partial sum of a Taylor series is called a *Taylor polynomial*. For example, the series $\sum_{k=0}^{\infty}(-1)^k \frac{x^{k+1}}{k+1}$ is a Taylor series, and the partial sum used in the previous example, $x-\frac{x^2}{2}+\frac{x^3}{3}-\frac{x^4}{4}+\frac{x^5}{5}$, is a fifth-degree Taylor polynomial.

The general form of a Taylor series is: $\sum_{k=0}^{\infty}\frac{f^{(k)}(a)}{k!}(x-a)^k$ where a is a known point on the function which the Taylor series is approximating. All of the examples so far have used $a = 0$ as the known point.*

The expression $f^{(k)}(a)$ requires some more explanation. The superscript k indicates which *derivative of the function* is being referred to. Derivatives will be discussed and developed in detail in Chapter 2, but for now think of them as a property belonging to a single point of a function. At each point of the function, the *slope* of the function is changing in a particular way. The first derivative, $f^{(1)}(a)$, is the slope of a function at a given point. The second derivative, $f^{(2)}(a)$, is the slope of the slope—that is, the change of the slope—at a given point. The third derivative, $f^{(3)}(a)$, is the change of the change of the slope at a given point, etc. You can see that the more derivatives you can compute at a given point, the more details you can learn about what is happening to the function near the given point. Each level of derivative at a given point is a clue about the "hidden" function nearby.

The Taylor series approach to approximating functions is specifically designed for situations where you know a lot about a single point and wish to extrapolate or approximate what the rest of the function is like.

*Taylor series with $a = 0$ are known collectively as Maclaurin series.

Of course, functions—particularly those describing real-world phenomena—come in many flavors, some better suited to Taylor series approximation than others. Some are not "infinitely differentiable"; they may have, say, first and second derivatives, but no third derivative. For these functions, you may only use a Taylor polynomial whose degree is less than or equal to the number of derivatives available. Some functions, although infinitely differentiable, are such that the derivatives at particular points are very poor clues about the rest of the function. The only absolute guarantee about Taylor series approximations is that they approximate the value of the function accurately at the given point. Some Taylor series approximate values well for all points, others only for a few points, and a fair number that manage to be accurate for the given point alone.

Moral: Use Taylor series approximation carefully—choosing the point at which you compute all the derivatives wisely and applying it only to functions that have sufficient derivatives so that you have a reasonable *interval of validity* for your approximation.

Although you haven't yet computed derivatives in this book, you can still compute Taylor polynomials. The HP 48 has a built-in function, TAYLR, that computes a Taylor polynomial (at $x = 0$) for a given function. In effect, it computes all the necessary derivatives of the function (at $x = 0$) and compiles the polynomial to whatever degree you specify.

Example: Compute the 6th degree Taylor polynomial for $1 + \sin^2 x$ using the built-in TAYLR command. Use it to approximate $1 + \sin^2(1.65)$.

1. Enter the function: ['][1][+][SIN][α][⬅][X][▶][y^x][2][ENTER].
2. Enter the variable to be used in the Taylor polynomial (also the independent variable in the function): ['][α][⬅][X][ENTER].
3. Enter the desired degree of the Taylor polynomial: [6][ENTER].
4. Compute the Taylor polynomial: [⬅][SYMBOLIC] TAYLR.

   Result: '1+x^2-8/4!*x^4+32/6!*x^6'
5. Make a copy of the polynomial and then approximate $1 + \sin^2(1.65)$: [ENTER][1][•][6][5]['][α][⬅][X][STO][EVAL].

   Result: 2.14868400625. (actual value: 1.99373988496)

Why is the approximation so bad? Recall on page 41 where the $\ln(1+x)$ function was plotted against its fifth-degree Taylor polynomial. Bad approximations occur when the point being evaluated lies outside the interval of validity for the approximation. And the interval of validity is always centered upon the point used to compute the Taylor polynomial coefficients. For the built-in TAYLR command, the interval of validity is always centered on $x=0$, and the farther from zero a value is, the greater the likelihood that its approximation will be bad.

Example: Use the Taylor polynomial calculated in the previous example to approximate $1 + \sin^2(0.65)$.

1. Drop the last result, to bring the Taylor polynomial to level 1: [←].
2. Store 0.65 in 'x' and evaluate the Taylor polynomial: [•][6][5]['] [α][←][X][STO][EVAL].

   Result: 1.36634986736 (Actual value: 1.36625058569)

Because 0.65 is much closer to zero than is 1.65, its approximation—which is "centered" at zero—is much better. So how do you get an accurate approximation of $1 + \sin^2(1.65)$, using a Taylor polynomial? Move the "center" of the approximation closer to 1.65.

Although the built-in TAYLR command doesn't allow you to do this, it is relatively simple with a program. TYLRa (see page 331) computes the Taylor polynomial for a given function centered around an arbitrary point *a* of your choosing. It takes the function being approximated from level 4, the independent variable from level 3, the order of the Taylor polynomial desired from level 2, and the point around which you want to center the approximation from level 1, and returns the Taylor polynomial to level 1.

Example: Find the sixth-degree Taylor polynomial centered at $x = 1.5$ for the function $1 + \sin^2 x$.

1. Enter the function to be approximated: ['][1][+][SIN][α][←][X][▶][y^x] [2][ENTER].
2. Enter the independent variable: ['][α][←][X][ENTER].
3. Enter the order of the polynomial desired: [6][ENTER].

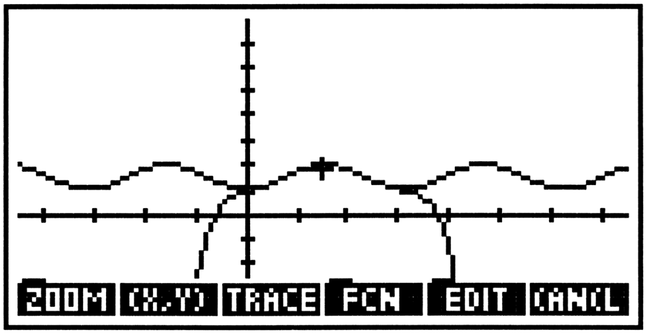
4. Enter the point around which the approximation is centered: [1][•][5] [ENTER].
5. Compute the Taylor polynomial using TYLRa: [α][α][T][Y][L][R][←][A] [ENTER] or [VAR] (then [NXT] or [←][PREV] as needed) TYLRA.

   Result (4 places): '1.9950+.1411*(x-1.5)
   -0.9900*(x-1.5)^2-.0941*(x-1.5)^3
   +.3300*(x-1.5)^4+.0188*(x-1.5)^5
   -.0440*(x-1.5)^6'

Computing the approximation around 1.5 instead of 0 should have the effect of shifting the interval of validity to the left (i.e. in the positive direction). To view the approximation and the function being approximated, and the interval of validity, you can use a procedure similar to the example on page 46 or you can use the PTAYL program (see page 317). PTAYL takes the Taylor polynomial from level 5, the function from level 4, the independent variable from level 3, the value around which the approximation is centered from level 2, and the value of the point being approximated from level 1. If there is no point being approximated, use same value you used in level 2 on level 1.

Example: Use PTAYL to visually depict the interval of validity for a sixth-degree Taylor polynomial approximation of the function, $1 + \sin^2 x$, centered around $x = 1.5$.

1. If the Taylor polynomial computed in the previous example is still on level 1, then move on to step 2. If it isn't, execute the previous example to compute it, and then return to step 2 of this example.
2. Enter the function: ['][1][+][SIN][α][←][X][▶][y^x][2][ENTER].
3. Enter the independent variable: ['][α][←][X][ENTER].
4. Enter the value around which the approximation is centered: [1][•][5] [ENTER].
5. Because you aren't interested in this example in estimating any one particular value, just enter a copy of previous value: [ENTER].
6. Execute PTAYL: [α][α][P][T][A][Y][L][ENTER] or [VAR] ([NXT] or [←][PREV] as needed) PTAYL.



7. Press TRACE (X,Y) and trace along the function with ◀ and ▶ to a region where the two graphs begin to separate. Then, jumping between the two graphs using ▲ and ▼, watch the value of the Y:-coordinate. Where it begins to differ significantly, you are leaving the interval of validity—to two decimal places, roughly $0.5 < x < 2.5$. Within that interval the polynomial approximates the function to two decimal places. (Of course, the decimal places necessary to validate an approximation may be more or less than two.)

Two other programs automate approximations, APROX and PAPROX (see pages 286 and 306). They each take five inputs: the function approximated on level 5, the independent variable on level 4, the order of the Taylor polynomial desired on level 3, the "centering" value on level 2, and the value to be estimated on level 1. APROX returns the Taylor polynomial used to level 3, the computed approximation on level 2, and estimate of the maximum error of the approximation to level 1. PAPROX first plots the function and its Taylor polynomial within a reasonable viewing range and then returns the same objects as does APROX.

Example: Use APROX to approximate the value of $1 + \sin^2 x$ at $x = 1.65$ at using a fourth-degree Taylor polynomial of $(x - 1.5)$.

1. Enter the function: ['] [1] [+] [SIN] [α] [←] [X] [▶] [y^x] [2] [ENTER].
2. Enter the independent variable: ['] [α] [←] [X] [ENTER].
3. Enter the degree of the Taylor polynomial: [4] [ENTER].
4. Enter the value of the independent variable around which the approximation is centered: [1] [•] [5] [ENTER].

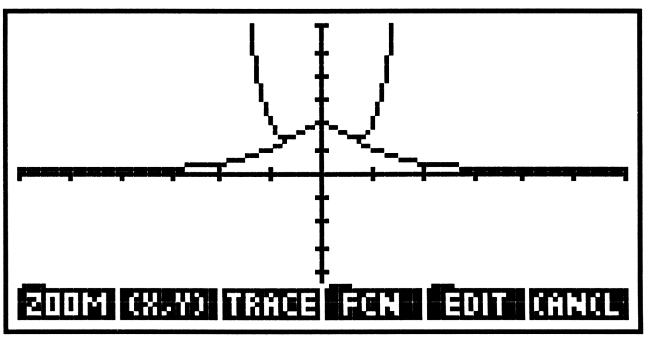
5. Enter the value you wish to approximate: [1][•][6][5][ENTER].
6. Execute APROX: [α][α][A][P][R][O][X][ENTER] or [VAR] (then [NXT] or [←][PREV] as needed) APROX. Result (to 3 places):

```
3:  '1.995+.141*(x-1.5)-.990*(x-1.5)^2-
     .094*(x-1.5)^3+.330*(x-1.5)^4
2:                1.994
1:                9.254E-7
```

The approximate value is 1.994, accurate to within less than 10^{-6}.

Example: Use PAPROX to approximate $\frac{2}{x^2+1}$ for x =0.5 using a fifth-degree Taylor approximation centered at x = 0.

1. Enter the function: ['][2][÷][←][()][α][←][X][y^x][2][+][1][ENTER].
2. Enter the independent variable: ['][α][←][X][ENTER].
3. Enter the degree of the Taylor polynomial desired: [5][ENTER].
4. Now the point around which the approximation is centered: [0][ENTER].
5. Enter the value you wish to compute: [•][5][ENTER].
6. Use PAPROX: [α][α][P][A][P][R][O][X][ENTER] or [VAR] ([NXT]) PAPRO.



The plot confirms that the target value (0.5) is within the interval of validity for the approximation.

7. Press [CANCEL] to return to the stack and see the approximation.

<u>Result</u>:
```
3:   '2-2*x^2+2*x^4'
2:             1.625
1:              .025
```

While PAPROX allows you to view a given Taylor approximation alongside the function it's approximating so that you can get a visual estimate of the interval of validity, it would be nice to be able to find the set of values for which any power series *converges*. This *interval of convergence* for a given power series,

$$\sum_{n=0}^{\infty} c_n (x-a)^n,$$

might be any one of three cases:

1. The power series converges within a *radius of convergence* (R) of $x = a$. That is, it converges for $a - R < x < a + R$, but diverges for other values of x. Note that the endpoints of the interval ($x = a - R$ and $x = a + R$) may or may not also be included in the interval and should be tested individually.
2. The power series converges only for the point around which the approximation is computed, $x = a$—nowhere else. The radius of convergence is 0.
3. The power series converges for all values of x. The radius of convergence is ∞.

The standard method of finding the interval of convergence for a power series involves applying the Root test to the series of coefficients, c_n, of the power series. That is, for the power series $\sum_{n=0}^{\infty} c_n (x-a)^n$, you must find $\lim_{n\to\infty} \sqrt[n]{|c_n|}$. If the limit ($L$) is finite and nonzero, then the radius of convergence is $1/L$ (Case 1). If the limit is infinite, then the radius of convergence is 0 (Case 2). If the limit is zero, then the radius of convergence is ∞ (Case 3).

The program CVINT (page 289) computes the radius of convergence for a given power series. It takes an expression representing the c_n portion of the power series from level 3, the independent variable from level 2, and the point a about which the power series is centered from level 1. It returns a string describing the approximate interval of convergence. Note that because CVINT computes limits as it determines the interval of convergence, the endpoints of the computed interval may be only approximate; you may need to apply common sense when testing the computed endpoints for inclusion or exclusion in the interval.

The following examples illustrate the proper use of CVINT.

Example: Find the interval of convergence for the series $\sum_{k=0}^{\infty} \frac{5^k}{k}(x-2)^k$.

1. Enter the c_n expression: ['][5][y^x][α][←][K][÷][α][←][K][ENTER].
2. Enter the index variable: ['][α][←][K][ENTER].
3. Now the point around which the power series is centered: [2][ENTER].
4. Compute the interval of convergence using CVINT: [α][α][C][V][I][N][T][ENTER] or [VAR] (then [NXT] or [←][PREV] as needed) [CVINT].

   Result: "1.8<x<2.2"

   The resulting interval is always open on both ends. You must test the endpoints separately by substitute each endpoint value for x and performing the set of convergence tests (SERX1, SERX2, SERX3).
5. Substitute 1.8 for x into the original power series and perform the integral test (SERX3) to determine convergence: ['][5][y^x][α][←][K][÷][α][←][K][×][←][()][•][2][+/−] [▶][y^x][α][←][K][ENTER]['][α][←][K][ENTER][VAR][SERX3].

   Result: { 0.0 0.0 0.0 0.0 }

   The integral test shows that the power series converges at $x = 1.8$.
6. Repeat step 5 using 2.2 for x: ['][5][y^x][α][←][K][÷][α][←][K][×][←][()][•][2][▶][y^x][α][←][K][ENTER]['][α][←][K][ENTER][SERX3].

   Result: { 2.3 0.0 0.0 0.0 }

   The integral test shows that the power series diverges at $x = 2.2$. (Because the integral test will normally show only increasing values—within reasonable precision—the decrease back to zero in this result list must be due to exceeding the limits of precision and thus can safely be ignored).

   Conclusion: The interval of convergence for the power series is:

   $$1.8 \le x < 2.2$$

Example: Find the interval of convergence for $\sum_{k=0}^{\infty} k(k+1)(x-1)^{2k}$.

1. Enter the c_n expression. Beware though, because the x-term is raised to the $2k$ power instead of the customary k power. To obtain a valid c_n expression, find the square root of the given c_n portion: ['] [α] [←] [K] [×] [←] [( )] [α] [←] [K] [+] [1] [ENTER] [√x].
2. Enter the index variable: ['] [α] [←] [K] [ENTER].
3. Now the point around which the power series is centered: [1] [ENTER].
4. Compute the interval of convergence using CVINT: [α] [α] [C] [V] [I] [N] [T] [ENTER] or [VAR] (then [NXT] or [←] [PREV] as needed) [CVINT].

   Result: "0<x<2"

   You can test the endpoints this time by inspection. Note that because the x-term is raised to the $2k$ power, both $x = 0$ and $x = 2$ will yield 1 as the value of the x-term, thus rendering the series equivalent to

   $\sum_{k=0}^{\infty} k(k+1)$ for both endpoints. This equivalent series clearly diverges. If you prefer, you can confirm this using either the limit comparison test (SERX2) or the integral test (SERX3); the root test is inconclusive.

   Conclusion: The interval of convergence for the power series is:

   $$0 < x < 2$$

Example: Find the interval of convergence for the power series: $\sum_{k=0}^{\infty} \frac{k!}{2}(x+1)^k$.

1. Enter the c_n expression: ['] [α] [⟲] [K] [α] [⟲] [DEL] [÷] [2] [ENTER].
2. Enter the index variable: ['] [α] [⟲] [K] [ENTER].
3. Enter the point around which the power series is centered: [1] [+/–] [ENTER].
4. Compute the interval of convergence using CVINT: [α] [α] [C] [V] [I] [N] [T] [ENTER] or [VAR] (then [NXT] or [⟲] [PREV] as needed) **CVINT**.

   Result: "x=-1"

   Conclusion: The power series converges nowhere but at $x = -1$.

Example: Find the interval of convergence for $\sum_{k=0}^{\infty} \frac{1}{2^{k!}}(x-2)^k$.

1. Enter the c_n expression: ['] [1] [÷] [2] [y^x] [α] [⟲] [K] [α] [⟲] [DEL] [ENTER].
2. Enter the index variable: ['] [α] [⟲] [K] [ENTER].
3. Now the point around which the power series is centered: [2] [ENTER].
4. Compute the interval of convergence using CVINT: [α] [α] [C] [V] [I] [N] [T] [ENTER] or [VAR] (then [NXT] or [⟲] [PREV] as needed) **CVINT**.

   Result: "All x"

   Conclusion: The power series converges for all x.

Estimating the Error of Approximation

Occasionally in the real-world application of power series, you will know a great deal about a specific point on a function without knowing the function itself. Indeed, this is a common use for power series approximations—as stand-ins for functions you cannot ascertain directly.

But how can you estimate how good your power series is as an approximation if you can't compare it with an actual function?

Taylor's Theorem, derived in most introductory calculus texts, provides a means for such an estimation of the error *if you can estimate the magnitude of the next higher order derivative than the order of the power series approximation.* For example, if you are using a fourth-degree Taylor approximation, you must be able to estimate the size of the fifth-derivative of the unknown function near the point around which you build the approximation.

Specifically, Taylor's Theorem implies that the error of a polynomial approximation (around point a) of order n for the value of a point b for a function f is:

$$\left|f(b)-p_n(b)\right|<\left|\frac{f^{(n+1)}(c)}{(n+1)!}(b-a)^{n+1}\right|$$

where $f^{(n+1)}(c)$ represents the maximum estimated value of the next higher order derivative (in the interval between a and b on the unknown function) than the order of the polynomial approximation.

The program TYLRERR (see page 331) assists you in estimating the error of approximation. As arguments, it takes the maximum estimated value of the next derivative ($f^{(n+1)}(c)$) from level 4, the degree of the approximation (n) from level 3, the point (a) around which the approximation was computed from level 2, and the point being approximated (b) from level 1. It then returns the maximum estimated error to level 1.

Example: For a given function approximated with a fourth-order Taylor series about $x = 1$, you estimate that the maximum value of the fifth derivative between $x = 1$ and $x = 3$ is 0.01. Find the maximum error using this approximation to compute the value of the function at $x = 3$.

1. Enter your estimate for the maximum value of the fifth derivative: [•][0][1][ENTER].
2. Enter the degree of the polynomial approximation: [4][ENTER].
3. Enter point around which the approximation was made: [1][ENTER].
4. Enter the point being approximated: [3][ENTER].
5. Compute the error approximation: [α][α][T][Y][L][R][E][R][R][ENTER] or [VAR] (then [NXT] or [⬅][PREV] as needed) TYLRE .

Result: 2.66666666667E-3

The maximum error using the fourth-order approximation to estimate the value of the unknown function at x=3 is ≈0.00267.

2. Differentiation and the Derivative

The Derivative and Differentiation: An Introduction

The *derivative* of a function $f(x)$ is itself a function that describes how fast $f(x)$ is changing at each point x. It is *derived* in the sense that it can't stand on its own—there must be an original function $f(x)$ to which it refers. *Differentiation* is the process of computing the derivative—either as a general expression or at a particular numerical point x. The name, *differentiation*, correctly implies that finding the derivative is based upon finding *differences*.

The rate at which a function changes is measured by its *slope*. As you recall from pre-calculus math courses, the slope of a function is measured by comparing the coordinates of two points on the function. The slope is the ratio of the *difference* of the vertical coordinates of the two points (Δy) and the *difference* of the horizontal coordinates of the two points (Δx).

Of course, the measurement $\Delta y/\Delta x$ can be interpreted only as the *average slope* of the function over the interval between the two points. But as you shrink this interval, the average slope describes an ever more precise region of the function. Finally, using the concept of limits, if you let the size of the interval approach zero, the average slope becomes the "slope-at-a -point"—instantaneous slope.

Thus the derivative is a function, $f'(x)$, that describes the instantaneous slope of each point in the domain of its referent function $f(x)$. Since the derivative uses the concept of limits, it requires that the referent function be continuous (at least in the interval being examined). So a function $f(x)$ is said to be *differentiable at a point, x_0*, provided that x_0 is in the domain of $f(x)$ and that this limit exists:

$$\lim_{x \to x_0} \frac{f(x) - f(x_0)}{x - x_0}$$

Or, because the quantity $(x - x_0)$ is more compactly written as Δx, the derivative function is commonly written as:

$$f'(x_0) = \lim_{\Delta x \to 0} \frac{f(x_0 + \Delta x) - f(x_0)}{\Delta x}$$

Roughly translated, this means that the derivative of a point is difference between the value of the function at the point and its value a "smidge" away from the point divided by the "smidge" as the size of the "smidge" approaches zero.

Derivatives on the HP 48

The HP 48 is programmed with the derivatives for all of its analytical functions and with the basic rules of differentiation (summarized below). Therefore it can find the derivative of any function composed entirely of those analytical functions—quite a large variety of functions, actually.

Here is a summary of what the HP 48 "knows" about derivatives. First and foremost, it is programmed with these derivative definitions:

Function	**Derivative**	**Function**	**Derivative**
x^r	rx^{r-1}		
a^x $(a>0, a\neq 1)$	$a^x \ln a$	$\log_a x$ $(a>0, a\neq 1)$	$\dfrac{1}{x \ln a}$
$\sin x$	$\cos x$	$\sin^{-1} x$	$\dfrac{1}{\sqrt{1-x^2}}$
$\cos x$	$-\sin x$	$\cos^{-1} x$	$-\dfrac{1}{\sqrt{1-x^2}}$
$\tan x$	$\dfrac{1}{\cos^2 x}$ (or $\sec^2 x$)	$\tan^{-1} x$	$\dfrac{1}{1+x^2}$
$\sinh x$	$\cosh x$	$\sinh^{-1} x$	$\dfrac{1}{\sqrt{1+x^2}}$
$\cosh x$	$\sinh x$	$\cosh^{-1} x$	$\dfrac{1}{\sqrt{x^2-1}}$
$\tanh x$	$\dfrac{1}{\cosh^2 x}$	$\tanh^{-1} x$	$\dfrac{1}{1-x^2}$

The HP 48 is also programmed with certain general rules for computing derivatives. If f and g are differentiable functions, k is a constant, and x_0 is a specific input in the domains of both f and g, then the following rules hold:

Linearity Properties:

$$(f+g)'(x_0) = f'(x_0) + g'(x_0)$$
$$(f-g)'(x_0) = f'(x_0) - g'(x_0)$$
$$(cf)'(x_0) = c \cdot f'(x_0)$$

Product Rule:

$$(fg)'(x_0) = f'(x_0)g(x_0) + f(x_0)g'(x_0)$$

Quotient Rule:

$$\left(\frac{f}{g}\right)'(x_0) = \frac{f'(x_0)g(x_0) - f(x_0)g'(x_0)}{g^2(x_0)}$$

Chain Rule:

$$(f \circ g)'(x_0) = f'(g(x_0)) \cdot g'(x_0)$$

Power Rule:

$$(f^n)'(x_0) = nf^{n-1}(x_0) \cdot f'(x_0)$$

Furthermore, the HP 48 can apply the derivative rules in several distinct ways, depending on the context in which you wish to see it. It can graph the derivative function along with the main function, compute the numeric derivative at a given point, or symbolically derive the derivative expression from the main function expression—either one step at a time ("step-wise" differentiation) or all at once.

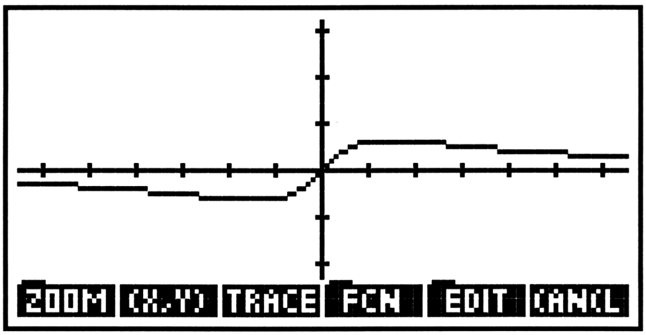
Look at some examples of each of these approaches in action ...

Graphically Displaying Derivatives

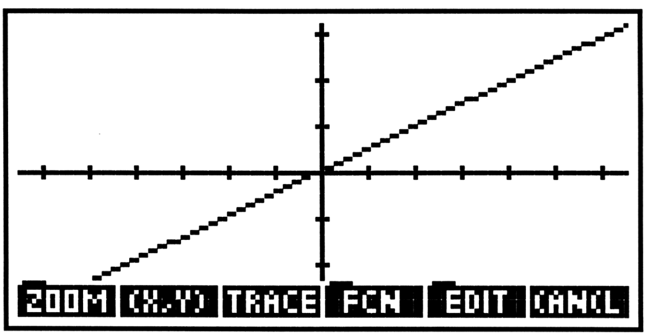
Functions that can be differentiated are "locally linear," a fact that can be ascertained best by graphing the function and zooming in on a region in question.

Example: Is $f(x) = \sin\left(\dfrac{2x}{x^2+2}\right)$ differentiable at $x = 0$?

1. Open the PLOT application, set the TYPE: to Function, and reset the plot parameters: [→] [PLOT] [▲] [α] [F] [DEL] [▼] [ENTER].
2. Highlight the EQ: field and enter the function expression: [▼] ['] [SIN] [2] [×] [α] [←] [X] [÷] [←] [()] [α] [←] [X] [y^x] [2] [+] [2] [ENTER].
3. Set the INDEP to x (lower-case) and press ERASE DRAW.



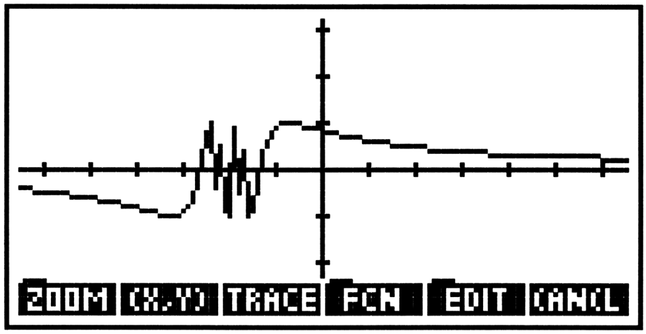
4. It *appears* smooth (i.e. continuous) and completely differentiable. But to test for differentiability at a point, zoom in tightly around the point and look for linearity: [▲] [◀] ZOOM BOXZ [▶] [▶] [▼] [▼] ZOOM.



The function is "locally linear"—thus differentiable—around $x = 0$.

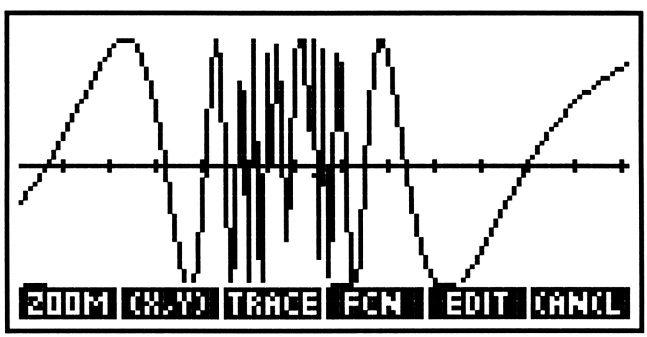
Example: Is $f(x) = \sin\left(\dfrac{2}{x+2}\right)$ differentiable at $x = -2$?

1. Return to the PLOT screen, and reset the plot parameters: [CANCEL] [DEL][▼][ENTER].
2. Highlight the **EQ:** field, and enter the function expression: [▲]['][SIN] [2][÷][←][()][α][←][X][+][2][ENTER].
3. Set the **INDEP** to **x** (lower-case) and press **ERASE** **DRAW**.



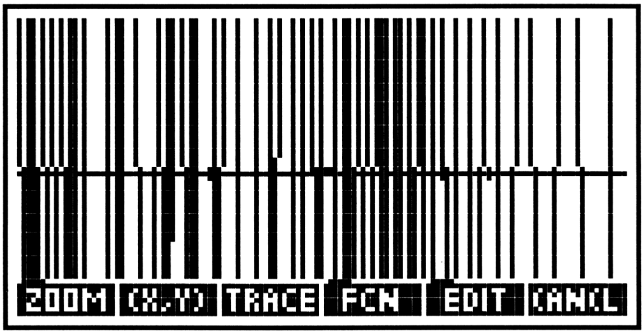
There appears to be some rapid oscillations in the neighborhood of the target point, $x = -2$, that needs investigation in more detail.

4. Move the cursor just above and to the left of the oscillations and press **ZOOM** **BOXZ**. Then draw the zoom-box to the right and just below the oscillations (using [▶] and [▼]) and then press **ZOOM**.



The oscillations are not straightening out as you zoom-in; instead, their chaotic behavior is increasing. You can confirm this by zooming-in on a box tightly constructed around the target point.

5. Draw the zoom-box so that it is about six pixels wide surrounding the middle of the oscillations and including the x-axis. Then zoom:

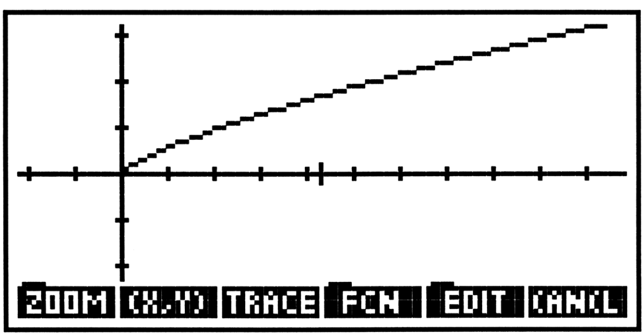


A vision of an undifferentiable function! At least, it's not differentiable in the vicinity of $x = -2$.

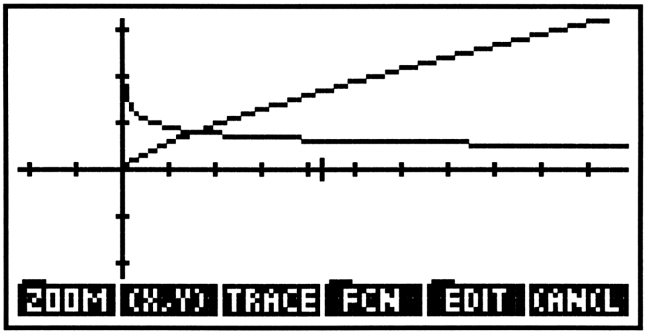
Once you can determine the differentiability of a function, you may then plot both the function and its derivative easily.

Example: Plot the function, $f(x) = x^{3/4}$ between $x = -1$ and $x = 5$, and then add the graph of its derivative. Use the default Y-VIEW parameters.

1. At the PLOT screen, reset the parameters: CANCEL DEL ▼ ENTER.
2. Highlight the EQ: field and enter the function: ▲ ' α ↰ X y^x ↰ ( ) 3 ÷ 4 ENTER.
3. Set the INDEP to x (lower-case) and change H-VIEW to −1 and 5.
4. Press ERASE DRAW.

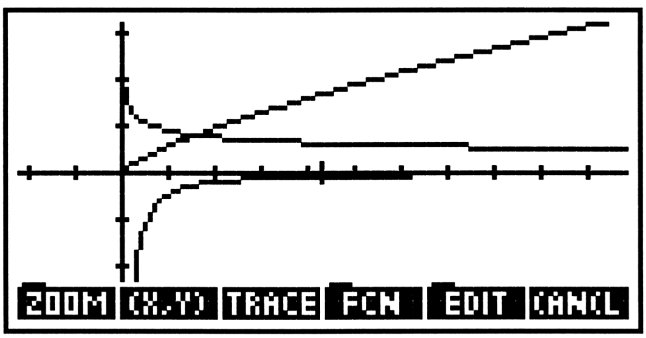


5. Add the derivative of the function to the plot: FCN (NXT) F'.



6. *Optional.* To view the expression for the derivative plotted, press (⮢) and then hold down (▼) (VIEW): '.75*x^-.25'

7. You can continue and plot the *second* derivative—the derivative of the derivative—by repeating step 5. (Note that, although this function has a second derivative, not all functions do: the first derivative must itself be differentiable.) Press FCN (NXT) F'.



8. (CANCEL) to return to the PLOT screen, and EDIT the EQ: field. It now lists three functions, the second and first derivatives and the original function: { '.75*-(.25*x^-1.25)' '.75*x^-.25' 'x^(3/4)' }

The first expression in the EQ: list is the current function for features such as TRACE, (X,Y), and FCN. But FCN (NXT) NXEQ rotates the order of the expressions, thus changing the current function.*

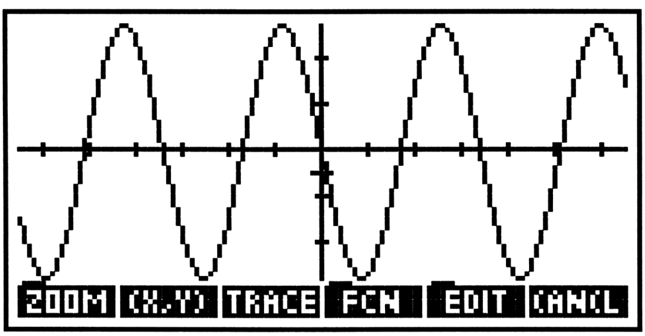
*The current equation can also be changed *temporarily* by pressing (▲) or (▼) while in TRACE mode. Upon leaving the Picture mode, the original order is reestablished in EQ.

This approach to plotting higher-order derivatives of a function results in all intermediate derivatives being plotted as well—a potentially confusing collection of curves in the display.

The program PLTDER (see page 313) plots just the derivative of a specified order —either with or without the current plot. It takes the function from level 4, the name of the independent variable from level 3, the order of the derivative from level 2, and a number indicating whether to erase the previous plot first (0 for "don't erase;" 1 for "erase") from level 1.

Example: Using PLTDER, plot the third derivative of $4\sin^2 x - 3\cos^2 x$.

1. Return to the stack and enter the function: [CANCEL][CANCEL]['][4][×][SIN][α][←][X][▶][y^x][2][−][3][×][COS][α][←][X][▶][y^x][2][ENTER].
2. Enter the independent variable: ['][α][←][X][ENTER].
3. Enter the order of derivative desired: [3][ENTER].
4. Enter 1 to erase the screen before plotting the derivative: [1][ENTER].
5. Open the PLOT application and reset the plot parameters to their defaults, then return to the stack: [→][PLOT][DEL][▼][ENTER][ENTER].
6. Plot the specified derivative: [α][α][P][L][T][D][E][R][ENTER] or [VAR] (then [NXT] or [←][PREV] as needed) [PLTDE].



Numerically Computing Derivatives

The slope of a function at a *particular* point is the function's derivative at that point. The HP 48 accomplishes this computation by first computing the symbolic derivative and then—if indicated—substituting the value of the point as the independent variable in the derived formula. To indicate a symbolic or numeric result when executing the "differentiate" command ([→][∂]), here are the options:

1. If flag -3 is *clear* and numeric values are stored for all variables in the function, then executing the command in *stack syntax* will return a numeric result. Stack syntax means entering the function on level 2, the variable of differentiation on level 1 and pressing [→][∂].
2. If flag -3 is *clear* and numeric values are stored for all variables in the function, then executing the command in *algebraic* syntax will return a symbolic result reflecting a *single* invocation—one "step"—of the chain rule for differentiation. Algebraic syntax means entering a single algebraic expression that includes the derivative function (eg. `'∂X(3*X^2+9)'`) and pressing [EVAL]. Each time you press [EVAL] you invoke the chain rule one more time and values may be substituted for some variables. Eventually a single number will be obtained—the same number you would have obtained immediately if you had used the stack syntax described above in option 1. This systematic approach to differentiation—*step-wise differentiation*—allows you to examine in detail each application of the chain rule.
3. If flag -3 is *clear* and at least one variable in the function has no value stored for it in the current path, then the HP 48 will return a symbolic result. This result will either reflect one invocation of the chain rule (if you used algebraic syntax) or the complete differentiation (if you used the stack syntax).
4. If flag -3 is *set*, then the HP 48 will find a numeric result if at all possible no matter how you execute the differentiate command and will indicate an error if it fails in its efforts.
5. Using the input screen of the Differentiate command (via [→][SYMBOLIC][▼] [ENTER]), you can explicitly request either the numeric or the complete symbolic result regardless of the current flag settings.
6. You can also use the input screen of the Differentiate command to perform a step-wise differentiation.

The following examples illustrate each of these options.

Example: *Option 1*. Find the numeric value of the derivative of $5x^2a + 3xa^2$ at $x = 8$. Let $a = 4$.

1. Make sure flag -3 is clear and store the appropriate values for x and a: [3][+/-][α][C][α][F][SPC][8]['][α][←][X][STO][4]['][α][←][A][STO].
2. Enter the function: ['][5][×][α][←][X][y^x][2][×][α][←][A][+][3][×][α][←][X][×][α][←][A][y^x][2][ENTER].
3. Enter the variable of differentiation: ['][α][←][X][ENTER].
4. Compute the derivative: [→][∂]. Result: `368`

Example: *Option 2*. Find the derivative of the function $5x^2a + 3xa^2$ at $x = 8$, using step-wise differentiation. Let $a = 4$.

1. Make sure flag -3 is clear and store the appropriate values for x and a: [3][+/-][α][C][α][F][SPC][8]['][α][←][X][STO][4]['][α][←][A][STO].
2. Create an algebraic expression for the derivative: ['][→][∂][α][←][X][←][()][5][×][α][←][X][y^x][2][×][α][←][A][+][3][×][α][←][X][×][α][←][A][y^x][2][ENTER].
3. Evaluate the derivative: [EVAL].

   Result: `'∂x(5*x^2*a)+∂x(3*x*a^2)'`

   The first application of the chain rule simply distributes the derivative function across the summation.

4. Evaluate again to see the next step in the differentiation: [EVAL].

   Result: `'∂x(5*x^2)*a+5*x^2*∂x(a)+(∂x(3*x)*a^2 +3*x*∂x(a^2))'`

5. Continue evaluating the result until you arrive at a number. In this case, it requires three more "steps:" [EVAL][EVAL][EVAL].

   Result: `368`—just as in the previous example.

Example: *Option 3*. Compute the symbolic derivative of $5x^2a+3xa^2$, where x and a remain variables.

1. Make sure flag -3 is clear and purge x and a to be sure that they remain symbolic in the result:* [3][+/-][α][C][α][F][ENTER][←][{ }][α][←][X][SPC] [α][←][A][ENTER][←][PURG].
2. Enter the function: ['][5][×][α][←][X][y^x][2][×][α][←][A][+][3][×][α][←] [X][×][α][←][A][y^x][2][ENTER].
3. Enter the variable of differentiation: ['][α][←][X][ENTER].
4. Compute the derivative: [→][∂].

Result: '5*(2*x)*a+3*a^2

Example: *Option 4.* Compute the numeric derivative of $5x^2a+3xa^2$ by repeating the previous example with flag -3 set.

1. Set flag -3: [3][+/-][α][S][α][F][ENTER].
2. Enter the function: ['][5][×][α][←][X][y^x][2][×][α][←][A][+][3][×][α][←] [X][×][α][←][A][y^x][2][ENTER].
3. Enter the variable of differentiation: ['][α][←][X][ENTER].
4. Compute the derivative: [→][∂].

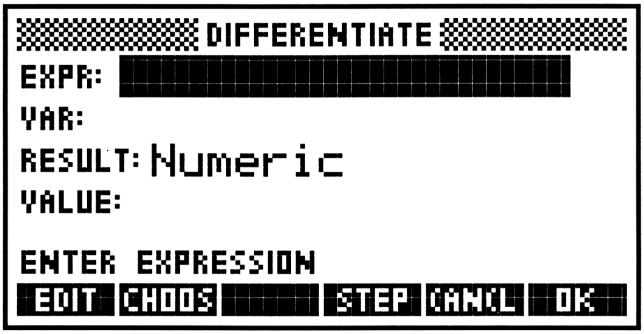
Result: Error: Undefined Name

This error message appears because a variable (or, as in this case, more than one variable) in the function is undefined, and yet the flag setting requires the HP 48 to find a numerical result.

*Remember that PURGing a variable removes it from the current directory only. To guarantee that a variable is symbolic, it must be purged from *all directories in the current path.* If the current directory is HOME as the example assumes, then PURGE is all you need. But if you're in a subdirectory, you may need to use the PGALL program (see page 313) to purge the variable throughout the current path.

Example: *Option 5.* Use the input screen to find the symbolic derivative of $5x^2a + 3xa^2$ at $x = 3$, where a remains undefined.

1. Open the DIFFERENTIATE input screen: [→][SYMBOLIC][▼][ENTER].



Note that the RESULT: field shows up as Numeric because the current setting of flag -3 is set (Numeric Results) from the previous example. Also, the EXPR: field may contain a function if you have used it previously in this or a related screen.

2. Enter the function into the EXPR: field: ['][5][×][α][←][X][y^x][2][×][α][←][A][+][3][×][α][←][X][×][α][←][A][y^x][2][ENTER].
3. Enter the variable of differentiation: ['][α][←][X][ENTER].
4. Because you wish to find a symbolic result, change the RESULT: field to Symbolic. Note that this change is a temporary one: it overrides the setting of flag -3 only in this one instance. After this computation, flag -3 will still be in its Numeric state (i.e. set): [+/–].
5. Store 3 in the variable x: [▲][NXT] CALC [3]['][α][←][X][STO] OK.
6. Compute the derivative: OK. Result: '30*a+3*a^2'

Example: *Option 6.* Use the **DIFFERENTIATE** input screen to perform a stepwise differentiation of $5x^2a + 3xa^2$ at $x = 3$, with a undefined.

1. Open the **DIFFERENTIATE** input screen: [→][SYMBOLIC][▼][ENTER].
2. The function should already show in the **EXPR:** field from the previous example, so enter the variable of differentiation: [▼]['][α] [←][X][ENTER].
3. Change the **RESULT:** field to `Symbolic`: [+/–].
4. The proper values are already stored in the variables from the previous example, so just compute the first "step" in the differentiation: **STEP**. Result: `'∂x(5*x^2*a)+∂x(3*x*a^2)'`.
5. Press [EVAL] to find the next step.

   Result: `Error: Undefined Name`

   What's this?!? Remember that your choice of `Symbolic` in the **RESULT:** field is temporary—good only for one operation. When you executed the second step of the differentiation, the flag -3 had returned to its set position (Numeric) and because a is undefined, the HP 48 returned an error.

   Moral: It is generally more convenient to leave flag -3 clear and just temporarily compute numeric results as needed, using `Numeric` in the input screen or [←][→NUM] from the stack.
6. Clear flag -3 and repeat this example. Now you will be able to evaluate the derivative to its completion.

Formal Derivatives

Finding a symbolic, or *formal*, derivative requires that the variable of differentiation be undefined. This is a problem if you don't want to purge the variable of differentiation—either you need to save the value, or it is simply too inconvenient to purge all such names in the current path.

One solution is to substitute a different (undefined) variable of differentiation, differentiate, then re-substitute the original variable into the result. This may seem as cumbersome as the original PURGing procedure, but it is more convenient as a program. The program, FDER, written by Bill Wickes in his book, *HP 48 Insights, Part II: Problem-Solving Resources* (included here with his permission—see page 294) does this. FDER takes the function from level 2 and the variable of differentiation from level 1 and returns the formal derivative to level 1.

Example: Use FDER to get the formal derivative (w/respect to x) of $5x^2a + 3xa^2$.

1. Enter the function: ['] [5] [×] [α] [←] [X] [y^x] [2] [×] [α] [←] [A] [+] [3] [×] [α] [←] [X] [×] [α] [←] [A] [y^x] [2] [ENTER].
2. Enter the variable of differentiation: ['] [α] [←] [X] [ENTER].
3. Execute FDER: [α] [α] [F] [D] [E] [R] [ENTER] or [VAR] (then [NXT] or [←] [PREV] as needed) [FDER]. Result: `'5*(2*x)*a+3*a^2'`. Note that this result is obtained even if x is defined and/or flag -3 is set.

Above, a was undefined and so just carried along in the differentiation. Naturally, if a is defined with a real number or an expression containing non-differentiation variables, then its value is substituted in the result. But if a should contain an expression with the differentiation variable, that will also affect the differentiation.

Example: Repeat the previous example after first storing `'SIN(x)'` in a.

1. Store sin x in a: ['] [SIN] [α] [←] [X] [ENTER] ['] [α] [←] [A] [STO].
2. Enter the function: ['] [5] [×] [α] [←] [X] [y^x] [2] [×] [α] [←] [A] [+] [3] [×] [α] [←] [X] [×] [α] [←] [A] [y^x] [2] [ENTER].
3. Enter the variable of differentiation: ['] [α] [←] [X] [ENTER].

4. Use FDER: [α][α][F][D][E][R][ENTER] or [VAR] ([NXT] or [←][PREV]) FDER.

   Result: '5*(2*x)*SIN(x)+5*x^2*COS(x)+(3*SIN(x)^2+3*x*(COS(x)*2*SIN(x)))'

   Now *a* has a big impact on the differentiation.

   Moral: Pay close attention to your non-differentiation variables, even when using FDER.

Angle Mode and Derivatives

One of the sneakier things that can affect your derivative computations for trigonometric functions is your angle mode.

Example: Purge *x* and compute the derivative of sin *x* first in Radian mode, then in Degree mode.

1. Purge *x* from the current path using PGALL (page 313): ['][α][←][X] [ENTER][α][α][P][G][A][L][L][ENTER].
2. Press [←][RAD], if necessary, to change to Radian mode.
3. Enter the function and the variable of differentiation: ['][SIN][α][←] [X][ENTER]['][α][←][X][ENTER].
4. Compute the derivative in Radian mode: [→][∂]. Result: 'COS(x)'
5. Compute the derivative in Degree mode: [←][RAD] ['][SIN][α][←][X] [ENTER]['][α][←][X][ENTER][→][∂]. Result: 'COS(x)*(π/180)'

   The results differ by a factor of π/180. Why? The sine function is defined primarily for radians, so a Degree-mode argument must first be converted to radians. Thus, SIN(x) becomes SIN(180*x/π) before the differentiation. Essentially, a trigonometric function in Degree mode is a different function from its Radian-mode equivalent and thus differentiates differently.

   Moral: Use Radian mode when differentiating (and integrating) with trigonometric functions.

Units and Derivatives

Many real-world problems using derivatives require physical units. However, when differentiating, physical units can be quite troublesome.

Example: Find the derivative with respect to V of $V^2h - gh^2$ when the variables contain the following values: $h = 5$ N, $g = 25$ cm.

1. Store the values with their units in the variables: [5][→][_][α][N]['][α][←][H][STO][2][5][→][_][α][←][C][α][←][M]['][α][←][G][STO].
2. Enter the function and differentiation variable: ['][α][V][y^x][2][×][α][←][H][−][α][←][G][×][α][←][H][y^x][2][ENTER]['][α][V][ENTER].
3. [→][∂]. Result: `+ Error: Inconsistent Units`

   What happened? Repeat the differentiation, but step-wise this time.
4. Enter the derivative expression: ['][∂][α][V][←][()][α][V][y^x][2][×][α][←][H][−][α][←][G][×][α][←][H][y^x][2][ENTER].
5. Begin evaluating: [EVAL]. Result: `'∂V(V^2*h)-∂V(g*h^2)'`
6. Then [EVAL][EVAL]. Result: `'∂V(V)*2*V^(2-1)*5_N-(0_N^2 +(25_cm)*(∂V(h)*2*h^(2-1)))'`

   The unit values are being inserted, and with the next evaluation, the HP 48 will apparently add Newtons to centimeters—not possible.
7. Confirm your suspicions: [EVAL].... Sure enough, that was the error.

The program `UDER` (see page 332) can find derivatives involving unit values. `UDER` has the same syntax as the standard stack version of the derivative.

Example: Find the derivative with respect to V of $V^2h - gh^2$ when the variables contain the following values: $h = 5$ N, $g = 25$ cm.

1. Enter the function and differentiation variable: ['][α][V][y^x][2][×][α][←][H][−][α][←][G][×][α][←][H][y^x][2][ENTER]['][α][V][ENTER].
2. Differentiate using `UDER`: [α][α][U][D][E][R][ENTER] or [VAR] ([NXT] or [←][PREV] as needed) **UDER**. Result: `'(10_N)*V'`

User-Defined Derivatives

Most of the HP 48's built-in functions that are continuous and differentiable (called *analytical functions*) have derivatives built in, too. That is, the HP 48 knows how to differentiate any expression composed exclusively of these built-in functions (i.e. most of the functions you commonly use). But you can include user-defined functions (or as-yet-undefined function names) in your work, too.

A quick review of user-defined functions. You define a user-defined function (UDF) by appending a parenthetical list of its arguments to its base name. For example, the %T (percent of total) function would be defined: `'%T(a,b)=100*b/a'`. The name of the UDF is `'%T(a,b)'` with the parentheses being used in the manner of standard function notation, such as *f(x)* or *g(x,y,z)*.*

Because the HP 48 knows no derivative with respect to *x* for `'%T(x,y)'`, when you request its derivative, it returns a *placeholder* name: `'der%T(x,y,1,0)'`. The `'der...'` prefix is reserved for use with these user-defined derivatives. Note that the derivative of a function must have exactly twice the number of arguments as the function itself—so that the derivative definition works with the chain-rule.

These placeholder names are indeed just names; as the user, you must define them as you would any UDF. For example, you might define the derivative of %T as: `'der%T(a,b,da,db)=(db/a-b/x^2*da)*100'`. (Notice how all four arguments figure in the definition.) Now, when the HP 48 is asked to find the derivative with respect to *x* of `'%T(x,y)'` it will evaluate the user-defined derivative and return: `'-(y/x^2*100)'`.

The use of UDF's as placeholding variable names (remember: `'der...'` variables are UDF's) can be quite awkward and visually confusing. So this book uses a different naming convention for placeholding names of derivatives. Instead of `'der%T(x,y,1,0)'`, for example (see above), this book would use `'δ%T.x'`. This more closely resembles standard book format, $\frac{d\%T}{dx}$ (which the HP 48 cannot use because it involves a division symbol where no division is intended).

*Note the critical difference between `'%T(a,b)'` and `'%T*(a,b)'`: the latter isn't a name at all, but rather the multiplication of the variable `'%T'` by the complex number `(a,b)`. The reverse problem—turning a multiplication into a UDF by accidentally omitting the * in front of the parentheses is an all-too-common error, also.

Implicit Differentiation

One of the common uses for placeholder derivatives is *implicit differentiation.* For example, if you were to differentiate the function, $(3x^3-4)y-2x+1$, with respect to *x*, using the standard method of differentiation, you would need to first solve explicitly for *y*: $y=\dfrac{2x-1}{3x^3-4}$. Then, once the independent (*x*) and dependent (*y*) variables have been separated on distinct sides of the equal sign, you would find the derivative normally: $\dfrac{dy}{dx}=-\dfrac{12x^3-9x^2+8}{(3x^3-4)^2}$

This procedure breaks down, however, when it is inconvenient (or impossible) to separate the independent and dependent variables (look at $4x^2+2xy-xy^3$, for example). This is where you can use *implicit differentiation.*

Standard differentiation treats variables other than the independent variable as *constants*. Thus, if you leave the dependent variables on the same side of the equation as the independent variable, standard differentiation will "mistreat" them—ignoring their status as variables altogether.

Implicit differentiation treats the dependent variables as *functions* of the independent variable, thus requiring that they be dealt with as variables during differentiation. Implicit differentiation (so named because it *implicitly* defines one or more dependent variables as functions of the independent variable) invokes the chain rule for differentiating functions imbedded within functions.

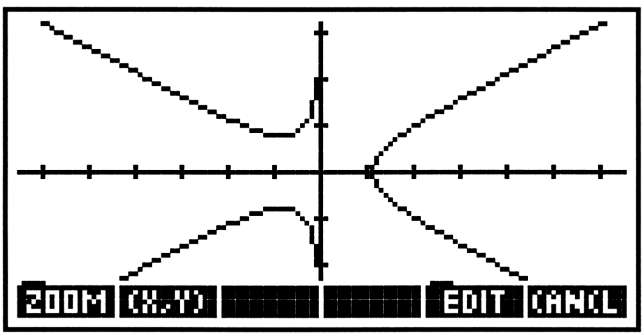
On the HP 48, there is a built-in means of distinguishing between a dependent variable to be treated as a constant during differentiation and one to be treated as a function of the independent variable. To make a variable act like a function, simply write it as such—a user-defined function. That is, instead of using `'y'` as the dependent variable, use `'y(x)'`.

Thus, differentiating $4x^2+2xy-xy^3$ with respect to *x* treats *y* as constant, yielding $8x+2y-y^3$. But differentiating $4x^2+2xy(x)-x(y(x))^3$ with respect to *x* invokes the chain rule: $8x+\left(2x\dfrac{dy}{dx}+2y\right)-\left(3xy^2\dfrac{dy}{dx}+y^3\right)$

Implicit differentiation is widely used in real-world computations because it easily handles *relations*—curves such as ellipses and spirals that don't obey the vertical line test for true functions. Relations have variables that mutually affect each other; no one variable can claim to be "independent" of the others.

Example: Plot $x^3 - 3xy^2 + y^3 - 1$, then find the slope of the curve at (2,-1).

1. Open the PLOT application and change the TYPE: field to Conic (Note that Conic may also be used to plot relations of two variables other than conic sections): [→][PLOT][▲][α][C].
2. Highlight the EQ: field, reset parameters, and enter the relation: [▼] [DEL][▼][ENTER]['][α][←][X][y^x][3][−][3][×][α][←][X][×][α][←][Y][y^x][2] [+][α][←][Y][y^x][3][−][1][ENTER].
3. Change the independent variable to x and the dependent variable to y: [α][←][X][ENTER][OPTS][▼][▶][α][←][Y][ENTER][ENTER].
4. Set H-VIEW to -6 6; plot: [6][+/−][ENTER][6][ENTER][ERASE][DRAW].



The relation clearly fails the vertical-line test; it is not a function.

5. Return to the stack, recall the relation (now in EQ) to level 1, and be sure that 'x' and 'y' are purged in the current path: [CANCEL] [CANCEL][α][E][α][Q][ENTER][←][{][α][←][X][SPC][α][←][Y][ENTER][1] [←][« »][α][α][P][G][A][L][L][ENTER][PRG][LIST][PROC][DOLIS].
6. To differentiate implicitly, you must replace each y in the relation with y(x). The ↓MATCH command allows you to substitute one symbolic expression for another. Enter a list on level 1 with the expression to be replaced ('y') as the first element and its replacement

('y(x)') as the second, then perform the substitution: [←][{ }]['][α][←][Y][▶]['][α][←][Y][←][( )][α][←][X][ENTER][←][SYMBOLIC][NXT][↓MAT].

Result: 2: 'x^3-3*x*y(x)^2+y(x)^3-1'
1: 1

7. The revised relation is on level 2 and a 1—a flag indicating a successful substitution—is on level 1. Drop the flag, enter the vari-able of differentiation, and differentiate: [DROP]['][α][←][X][ENTER][→][∂].

   Result: '3*x^2-(3*y(x)^2+3*x*(dery(x,1)*2*y(x)))+dery(x,1)*3*y(x)^2'

8. The 'dery(x,1)' is a user-defined derivative (see page 71), a placeholder variable representing $\frac{dy}{dx}$ (the slope of the relation), created by the HP 48 because 'y(x)' is not defined. To find the slope at the given point, just solve for the 'dery(x,1)' and substitute values for x and y. Normally you would use QUAD to algebraically solve for an unknown variable, but not when undefined functions are present—and 'dery(x,1)' and 'y(x)' are both undefined. So first substitute (using ↓MATCH again) with actual variable names. You can use any allowable name, but this book names derivatives with a "∂" ([α][→][D]) followed by the name of the function, then a period ("."), then the differentiation variable. Therefore 'dery(x,1)' becomes '∂y.x' (and 'y(x)' becomes 'y'): [←][{ }]['][α][←][D][α][←][E][α][←][R][α][←][Y][←][( )][α][←][X][←][,][1][▶][▶]['][α][→][D][α][←][Y][•][α][←][X][ENTER][↓MAT][DROP][←][{ }]['][α][←][Y][←][( )][α][←][X][▶][▶]['][α][←][Y][ENTER][↓MAT][DROP].

   Result: '3*x^2-(3*y^2+3*x*(∂y.x*2*y))+∂y.x*3*y^2'

9. Find the general slope by solving for '∂y.x' via QUAD—now that the expression contains no UDF's: ['][α][→][D][α][←][Y][•][α][←][X][ENTER][←][PREV][QUAD]. Result: '∂y.x=-((3*x^2-3*y^2)/(-(3*x*(2*y))+3*y^2))'

10. Now substitute the coordinates of the given point for x and y to get a numerical value: [2]['][α][←][X][STO][1][+/-]['][α][←][Y][STO][EVAL]. Result: '∂y.x=-.6'. The slope of the relation at (2,-1) is -0.6.

The program IMPS (see page 295) automates the involved process of implicit differentiation. IMPS takes the implicit relation from level 2 and a list of the variables in the relation from level 1. The first variable in the level 1 list must be the variable of differentiation.

Example: Find the implicit derivative (with respect to x) of $x^3 - 3xy^2 + y^3 - 1$, using IMPS.

1. Enter the relation: ['] [α] [←] [X] [y^x] [3] [−] [3] [×] [α] [←] [X] [×] [α] [←] [Y] [y^x] [3] [−] [1] [ENTER].
2. Enter the list of variables, placing the variable of differentiation, x, first: [←] [{ }] [α] [←] [X] [SPC] [α] [←] [Y] [ENTER].
3. Compute the formal implicit derivative using IMPS: [α] [α] [I] [M] [P] [→] [D] [ENTER] or [VAR] (then [NXT] or [←] [PREV] as needed) [IMPS].

   Result: `'3*x^2-(3*y^2+3*x*(∂y.x*2*y))+∂y.x*3*y^2'`

IMPS also makes it easy to implicitly differentiate a relation with respect to a variable that doesn't obviously figure in the relation. This is commonly the case when working with related rates problems, discussed in greater detail beginning on page 112. Look at an example:

Example: If V and n are functions of time, t, differentiate $V^3 = kn^2 - 3n$ with respect to t.

1. Enter the relation: ['] [α] [V] [y^x] [3] [←] [=] [α] [←] [K] [×] [α] [←] [N] [y^x] [2] [−] [3] [×] [α] [←] [N] [ENTER].
2. Enter the list of variables, adding the variable of differentiation, t, as the first element: [←] [{ }] [α] [←] [T] [SPC] [α] [V] [SPC] [α] [←] [N] [ENTER].
3. Compute the formal implicit derivative using IMPS: [α] [α] [I] [M] [P] [→] [D] [ENTER] or [VAR] (then [NXT] or [←] [PREV] as needed) [IMPS].

   Result: `'∂V.t*3*V^2=k*(∂n.t*2*n)-3*∂n.t'`

   Or, in standard notation: $3V^2 \dfrac{dV}{dt} = 2kn\dfrac{dn}{dt} - 3\dfrac{dn}{dt}$

Derivatives of Polynomials

For the purposes here, a *polynomial* is a function of a single variable of the form, $P(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0$, where n is a positive integer. The real numbers a_n, a_{n-1}, a_{n-2}, ..., a_1, a_0 are the coefficients, and if $a_n \neq 0$, the polynomial is said to have degree n. The ratio of two polynomials is a *rational fraction*.

Because polynomials have a single variable, they differ from one another only in their coefficients, allowing the HP 48 to compute with a polynomial more rapidly that with many other functions, using a *vector* of its coefficients. For example, $2x^5 - 3x^4 + x^3 + 6x^2 - 18x + 11$ would be `[ 2 -3 1 6 -18 11 ]`; and $2x^5 + x^3 + 11$ would be `[2 0 1 0 0 11 ]`. There is a set of programs designed to simplify operations on polynomials, named here with their page numbers:*

`PADD` (page 305): Add two polynomials.
`PSUB` (page 317): Subtract two polynomials.
`PMULT` (page 313): Multiply two polynomials.
`PPOWER` (page 315): Raise a polynomial to a positive integral power
`PDIV` (page 308): Divide (Euclidean) of two polynomials, M and N, resulting in Q (quotient) and R (remainder) polynomials such that $M = NQ + R$. Note that polynomial division doesn't always yield a polynomial.
`PDIV2` (page 308): Divide two polynomials after eliminating common factors.
`PCONV` (page 307): Convert symbolic expression (1 variable) to polynomial.
`P→SYM` (page 317): Convert a polynomial in vector form to symbolic form.
`RF→S` (page 320): Convert a rational fraction to symbolic form.
`S→RF` (page 328): Convert symbolic expression (1 variable) to rational fraction.
`PDER` (page 307): Computes the derivative of a polynomial.
`PF` (page 309): Computes the partial fraction expansion of a rational fraction.
`PFACT` (page 310): Convert polynomial into factors with integral coefficients.
`PPROD` (page 315): Finds the derivative of the product of a list of polynomials.
`PQUOT` (page 316): Find derivative of rational fraction via quotient rule.
`PREDUCE` (page 316): Reduce polynomial coefficients to lowest integral values.
`REMNDR` (page 319): Convert results of `PDIV` or `PDIV2` to symbolic result.

*Many were written for the book *Algebra and Pre-Calculus on the HP 48G/GX* and are treated in more detail there.

As an example of the efficiency of these programs, compare using the built-in derivative with PDER to find the derivative of a polynomial.

Example: Find the derivative of $x^6 + 4x^5 - 3x^3 + 8x^2 - 15$. Use the built-in derivative first, then PDER.

1a. Enter the symbolic polynomial: ['] [α] [←] [X] [yˣ] [6] [+] [4] [×] [α] [←] [X] [yˣ] [5] [−] [3] [×] [α] [←] [X] [yˣ] [3] [+] [8] [×] [α] [←] [X] [yˣ] [2] [−] [1] [5] [ENTER].

1b. Enter the variable of differentiation and make an extra copy: ['] [α] [←] [X] [ENTER] [ENTER].

1c. Purge the variable of differentiation from the current directory and then differentiate: [←] [PURG] [→] [∂].

Result 1: '6*x^5+4*(5*x^4)-3*(3*x^2)+8*(2*x)'

2a. Enter the polynomial as a vector of its coefficients. Don't forget to include zeroes for the "missing" x^4 and x^1 terms: [←] [[ ]] [1] [SPC] [4] [SPC] [0] [SPC] [3] [+/−] [SPC] [8] [SPC] [0] [SPC] [1] [5] [+/−] [ENTER].

2b. Compute the derivative using PDER: [α] [α] [P] [D] [E] [R] [ENTER] or [VAR] (then [NXT] or [←] [PREV] as needed) [PDER].

Result 2: [6 20 0 -9 16 0] or $6x^5 + 20x^4 - 9x^2 + 16x$

2c. *Optional.* Convert the polynomial result back to a symbolic using the P→SYM program: ['] [α] [←] [X] [α] [α] [P] [→] [→] [S] [Y] [M] [ENTER].

Result: '6*x^5+20*x^4-9*x^2+16*x'

Notice that the vector form for polynomials is not only faster to enter but faster to compute with, as well. The other advantage it offers is evident when you're working with polynomials in situations where you can apply the product or quotient rules....

Example: Find the derivative of $(x^2+5)^3(x+1)(2x-9)$, first using the built-in derivative, then using the PPROD program.

1. Enter the symbolic expression: ['] [←] [()] [α] [←] [X] [y^x] [2] [+] [5] [▶] [y^x] [3] [×] [←] [()] [α] [←] [X] [+] [1] [▶] [×] [←] [()] [2] [×] [α] [←] [X] [−] [9] [ENTER].
2. Enter the variable of differentiation: ['] [α] [←] [X] [ENTER].
3. Differentiate: [→] [∂].

   Result: '(2*x*3*(x^2+5)^2*(x+1)+(x^2+5)^3)*(2*x-9)+(x^2+5)^3*(x+1)*2'

A. Create a list of the polynomial factors. You'll probably want to use PPOWER to compute the cube of the polynomial [1 0 5] (i.e. x^2+5): [←] [[ ]] [1] [SPC] [0] [SPC] [5] [ENTER] [←] [[ ]] [3] [ENTER] [α] [α] [P] [P] [O] [W] [E] [R] [ENTER] [←] [[ ]] [1] [SPC] [1] [ENTER] [←] [[ ]] [2] [SPC] [9] [+/−] [ENTER] [PRG] LIST [3] →LIST.

B. Differentiate using PPROD: [α] [α] [P] [P] [R] [O] [D] [ENTER] or [VAR] (then [NXT] or [←] [PREV] as needed) PPROD.

   Result: [16 -49 126 -525 60 -1575 -850 -875]

C. *Optional*. Convert the polynomial result back to a symbolic using the P→SYM program: ['] [α] [←] [X] [ENTER] P→SYM.

   Result: '16*x^7-49*x^6+126*x^5-525*x^4+60*x^3-1575*x^2-850*x-875'.

The result comes more quickly in the latter case, and it's in a simpler form. The HP 48 multiplies (or divides) two factors, then multiplies (or divides) the result by the third factor, etc., thereby creating an expression with nested parentheses *before* being differentiated. Because of this nesting, the built-in command often yields complicated (nay, ugly) results. However, notice that the built-in result retains evidence of the product rule at work (it's built into the command), whereas the PPROD result does not. This may be important to you in a particular context.

To see the nesting problem in its full glory, look at the quotient rule.

Example: Use the differentiation command to compute the derivative of

$$\frac{\left(x^2+1\right)^3(2x-5)^2}{\left(x^2+5\right)^2}.$$

1. Enter the function (using the EquationWriter): [←][EQUATION][▲][←][( )][α][←][X][y^x][2][▶][+][1][▶][y^x][3][▶][←][( )][2][α][←][X][−][5][▶][y^x][2][▶][▶][←][( )][α][←][X][y^x][2][▶][+][5][▶][y^x][2][ENTER].
2. Enter the variable of differentiation: ['][α][←][X][ENTER].
3. Differentiate: [→][∂].

   Result: `'(2*x*3*(x^2+1)^2*(2*x-5)^2+(x^2+1)^3*(4*(2*x-5)))/(x^2+5)^2-(x^2+1)^3*(2*x-5)^2*(2*x*2*(x^2+5))/(x^2+5)^2^2'`

There are two reasonable alternatives to the standard method, which you may find helpful in returning equivalent—but more readable—results:

1. Using *polynomial shortcuts*, you can fully expand the numerator and denominator factors, so that you have no exponents applied to groups of terms, only to individual terms. Then, using a single application of the quotient rule—via the program `PQUOT`—compute the result.
2. Use *logarithmic differentiation*. Find the natural logarithm of the original function, differentiate the result, and then multiply by the original function. This result is a series of additive terms. (Logarithmic differentiation, discussed next, isn't limited to polynomial functions and rational fractions.)

Look at the first method, because it applies to polynomials and rational fractions. It makes use of two programs, `S→RF` and `PQUOT`.

Because expanding polynomials in their symbolic form is notoriously slow on the HP 48, the program `S→RF` (see page 328) converts the symbolic expression (level 1) composed of only polynomials in both numerator and denominator to an expanded numerator polynomial, given as an array of its coefficients (level 2) and an expanded denominator polynomial (level 1), also given as an array.

PQUOT (see page 316) uses the quotient rule for differentiating a rational fraction whose numerator (level 2) and denominator (level 1) are arrays of coefficients.

Example: Find the derivative of $\dfrac{(x^2+1)^3(2x-5)^2}{(x^2+5)^2}$.

1. Enter the function: [←][EQUATION] [▲] [←][( )] [α][←][X] [y^x] [2] [▶] [+] [1] [▶] [y^x] [3] [▶] [←][( )] [2] [α][←][X] [−] [5] [▶] [y^x] [2] [▶] [▶] [←][( )] [α][←][X] [y^x] [2] [▶] [+] [5] [▶] [y^x] [2] [ENTER].
2. Expand the expression to a simple rational fraction: [α][α][S][→][→] [R][F][ENTER] or [VAR] (then [NXT] or [←][PREV] as needed) [S→RF].

   Result:
   ```
   2: [ 4 -20 37 -60 87 -60 79 -20 25 ]
   1:      [ 1 0 10 0 25 ]
   ```
   This represents:

   $$\frac{4x^8-20x^7+37x^6-60x^5+87x^4-60x^3+79x^2-20x+25}{x^4+10x^2+25}$$

3. Find the derivative of the rational fraction using the quotient rule: [α][α][P][Q][U][O][T][ENTER] or [VAR] ([NXT] or [←][PREV]) [PQUO].

   Result:
   ```
   2: [ 16 -60 234 -760 1110 -1440 1582
        -840 690 -100 ]
   1: [ 1 0 15 0 75 0 125 ]
   ```
   This represents:

$$\frac{16x^9-60x^8+234x^7-760x^6+1110x^5-1440x^4+1582x^3-840x^2+690x-100}{x^6+15x^4+75x^2+125}$$

4. *Optional.* You can also factor the numerator and denominator, using PFACT (see page 310). Swap the numerator into level 1 and factor it; swap again and factor the denominator: [SWAP][α][α][P][F][A][C][T] [ENTER] or [VAR] ([NXT] or [←][PREV] as needed) [PFACT] [SWAP] [PFACT].

   Result:
   ```
   2: { [ 2 ] [ 2 -5 ] [ 1 0 1 ]
        [ 1 0 1 ] [ 4 -5 38 -65 10 ] }
   1: { [ 1 0 5 ] [ 1 0 5 ] [ 1 0 5 ] }
   ```

   So the simplified derivative is

   $$\frac{2(2x-5)(x^2+1)^2(4x^4-5x^3+38x^2-65x+10)}{(x^2+5)^3}.$$

Logarithmic Differentiation

Logarithmic differentiation is often a good method for differentiating a complicated product of the form $g(x)=\dfrac{h_1(x)h_2(x)\cdots h_n(x)}{j_1(x)j_2(x)\cdots j_m(x)}$, where h_n and j_m are functions of x on the numerator and denominator, respectively. Logarithmic differentiation requires that you find the natural logarithm of both sides of the equation before differentiating. Thus, finding the natural log yields:

$$\ln|g(x)| = \ln|h_1(x)| + \ln|h_2(x)| + \cdots + \ln|h_n(x)| - \left(\ln|j_1(x)| + \ln|j_2(x)| + \cdots + \ln|j_m(x)|\right)$$

Then differentiating and solving for the derivative function yields:

$$g'(x) = g(x)\left[\frac{h_1'(x)}{h_1(x)} + \frac{h_2'(x)}{h_2(x)} + \cdots + \frac{h_n'(x)}{h_n(x)} - \left(\frac{j_1'(x)}{j_1(x)} + \frac{j_2'(x)}{j_2(x)} + \cdots + \frac{j_n'(x)}{j_n(x)}\right)\right]$$

The program `LNS` (see page 301) implements logarithmic differentiation for you. It takes a list of the symbolic numerator factors (h_1, h_2, ..., h_n) from level 3, a list of the symbolic denominator factors (j_1, j_2, ..., j_m) from level 2, and the variable of differentiation from level 1. `LNS` returns to level 1 a symbolic expression representing $g'(x)$ in the form shown above.

Example: Use logarithmic differentiation to find the derivative of

$$\frac{\left(x^2+1\right)^3(2x-5)^2}{\left(x^2+5\right)^2}.$$

1. Enter a list of the numerator factors: [↰][{ }]['][↰][( )][α][↰][X][yˣ][2] [+][1][▶][yˣ][3][▶]['][↰][( )][2][X][α][↰][X][−][5][▶][yˣ][2][ENTER].
2. Enter a list of the denominator factors: [↰][{ }][↰][( )]['][α][↰][X][yˣ][2] [+][5][▶][yˣ][2][ENTER].
3. Enter the variable of differentiation: ['][α][↰][X][ENTER].
4. Perform the logarithmic differentiation using `LNS`: [α][α][L][N][↱][D] [ENTER] or [VAR] (then [NXT] or [↰][PREV] as needed) [LNS].

   <u>Result</u>: `'(6/(1+x^2)*x+4/(-5+2*x)-4/(5+x^2)*x)*(-5+2*x)^2*(1+x^2)^3*(5+x^2)^-2'`

Derivatives of Polar and Parametric Functions

The standard representation of a function, such as $y = 4x^2 + \sin 2x$, implies that the function's output value, y, depends upon the input value, x. Such "dependence" can be misleading, however, whenever the function describes a situation where the variables involved are actually independent of one another.

For example, the function $y = -x^2/64$ adequately describes the curve a rock takes as it is thrown horizontally off a cliff at 32 ft/sec. When the rock is horizontally x feet from the cliff, it is y feet below its starting point. However, despite the appearance that y depends on x, the horizontal and vertical motions are actually independent of each other.

The *parametric representation* of the function emphasizes the true independence of the two variables by making each dependent on a third value—a *parameter*. Thus, parametrically, the function becomes $x = 32t$; $y = -16t^2$, where t is the time (in seconds) after the throw.

Many real-world situations are best represented parametrically, but the parametric form makes computing the slope, $\frac{dy}{dx}$, of a curve a bit more difficult than for curves in standard form.*

The strategy for parametrically-described curves is to compute the derivatives of each of the parameter definitions with respect to the parameter t, then solve for $\frac{dy}{dx}$. That is, because of the chain rule, $\frac{dy}{dx} = \frac{dy\,/\,dt}{dx\,/\,dt}$.

*In Chapter 6, the program VDER (see page 332 for the program listing) performs a related task for a vector-valued function. Vector representation and parametric representations are closely linked.

The HP 48's built-in differentiation tools work with functions in standard form only. However, the program PARAS (see page 306) computes the slope of a function described parametrically at a given value of the parameter, t. It takes a list of parameter definitions—i.e. functions of t—from level 2 and a value of t from level 1 and returns the list of symbolic derivatives (with respect to t) of each of the parameter definitions to level 3, the symbolic expression for the slope in terms of t on level 2, and the numeric slope at the given value of t to level 1. Note that PARAS requires that you use 't' (lowercase) as the parameter.

Example: Find the slope of the curve described by $x = 32t \quad y = -16t^2$ when $t = 3.5$ seconds.

1. Enter the list of parameter definitions: [←][{ }]['][3][2][×][α][←][T][▶] ['][1][6][+/-][×][α][←][T][y^x][2][ENTER].
2. Enter the value for the parameter: [3][•][5][ENTER].
3. Compute the slope using PARAS: [α][α][P][A][R][A][→][D][ENTER] or [VAR] (then [NXT] or [←][PREV] as needed) [PARAS].

<u>Result</u>:

```
3:    { 32 '-(32*t)' }
2:                '-t'
1:                -3.5
```

The slope at $t = 3.5$ is -3.5 and the symbolic slope expression is $-t$.

One special form of parametric representation is the expression of a function in the *polar* coordinate system. A function $f(x)$ in the rectangular coordinate system can be parametrized as a function in the polar coordinate system, $r(\theta)$ where the polar angle, θ, is the parameter, and x and y depend upon θ as follows:

$$x = r(\theta)\cos\theta; \qquad\qquad y = r(\theta)\sin\theta$$

where $r(\theta)$ is the polar function. That is, a polar function is equivalent to a rectangular function expressed parametrically via the relationship shown above.

Example: To express the polar function $r(\theta) = 2\sin(\theta)$ as a parametric form of a rectangular function, replace $2\sin(\theta)$ for $r(\theta)$ in the parameter definitions above: $x = 2\sin\theta\cos\theta;$ $\qquad y = 2\sin^2\theta$

Finding the slope of a polar function can be achieved using the same chain-rule approach as for parametric functions shown above:

$$\frac{dy}{dx} = \frac{dy/d\theta}{dx/d\theta} = \frac{\dfrac{dr}{d\theta}\sin\theta + r(\theta)\cos\theta}{\dfrac{dr}{d\theta}\cos\theta - r(\theta)\sin\theta}$$

The program, POLδ (see page 314), computes the symbolic and numerical slope of a polar function for a given value of the function. It takes the function expression using 'θ' from level 2 and the given value of θ from level 1 and returns the expression for the slope (level 2) and the numerical slope at the value (level 1).

Example: Find the slope of the polar function, $r = 1 + 2\sin\theta$, at $\theta = \dfrac{\pi}{3}$.

1. Enter the polar function: ['] [1] [+] [2] [×] [SIN] [α] [↱] [F] [ENTER].
2. Enter the value of the polar angle: ['] [↰] [π] [÷] [3] [ENTER].
3. In Rad mode, find the slope using POLδ: ([↰] [RAD], if necessary) [α] [α] [P] [O] [L] [↱] [D] [ENTER] or [VAR] ([NXT] or [↰] [PREV]) [POLδ].

<u>Result</u>:

```
2:  '((1+2*SIN(θ))*COS(θ)+SIN(θ)*(2*
    COS(θ)))/(COS(θ)*(2*COS(θ))-(1+
    2*SIN(θ))*SIN(θ))'
1:                   -1.19615242269
```

Notes

3. Applications of the Derivative

This chapter contains many examples of the use of derivatives in solving many important types of problems, including:

- Computing the rate of change "at the margin" of a function.
- Determining a function's critical points.
- Determining the maximum or minimum of a function over a given interval.
- Finding the line tangent to a curve at a particular point.
- Finding the angle between two curves at their point of intersection.
- Computing the rate of change of one quantity from the rate of change of a physically related quantity.

Marginal Analysis

The term *marginal* is often heard in financial analyses: marginal rate of return, marginal profit, marginal tax rate, increasing marginal costs, etc. "Margin" is another term for the slope of a function, so the phrase "marginal profit," for example, refers to the slope of the profit function (i.e. profit as a function of production quantity) at any given point (i.e. level of production). "What," asks the manufacturer, "would be the effect on my profit of producing one more unit than I'm now producing? That is, what is the effect of producing the *next* unit, the unit that's on the *margin* of my current production?"

For example, suppose a stereo manufacturer determines that its cost per stereo is $C(x) = 3000 + 20x$. This would reflect $3000 of fixed overhead plus $20 per stereo produced. Further, the manufacturer computes that its average revenue per stereo is $R(x) = 1000x - x^2$. The stereo manufacturer currently manufactures and sells 500 stereos. Would an expansion of production to 501 stereos be profitable? In other words, what is the *marginal profit* of the 501st stereo?

Profit is simply revenues minus costs, so the profit curve is:

$$P(x) = R(x) - C(x) = 1000x - x^2 - (3000 + 20x) = -x^2 + 980x - 3000$$

The manager's question can be answered by determining the slope of the profit curve at $x = 500$, a perfect application for the derivative....

Example: Find the marginal profit at $x = 500$ of a process with this profit curve:

$$P(x) = -x^2 + 980x - 3000$$

1. Enter the profit function: ['] [+/−] [α] [←] [X] [y^x] [2] [+] [9] [8] [0] [×] [α] [←] [X] [−] [3] [0] [0] [0] [ENTER].
2. Enter the independent variable: ['] [α] [←] [X] [ENTER].
3. Store 500 in 'x': [5] [0] [0] [ENTER] [←] [STACK] [OVER] [STO].
4. Compute the marginal profit by computing the derivative: [→] [∂].

   Result: -20

   The marginal profit is negative! This means that manufacturing the next additional unit (the 501st) actually subtracts \$20 from profit.

Example: Repeat the previous example for $x = 450$.

1. Enter the profit function: ['] [+/−] [α] [←] [X] [y^x] [2] [+] [9] [8] [0] [×] [α] [←] [X] [−] [3] [0] [0] [0] [ENTER].
2. Enter the independent variable: ['] [α] [←] [X] [ENTER].
3. Store 450 in 'x': [4] [5] [0] [ENTER] [OVER] [STO].
4. Compute the marginal profit by computing the derivative: [→] [∂].

   Result: 80

   This time, manufacturing the next additional unit (the 451st) adds \$80 to the profit.

Thus, although the profit function itself doesn't change, marginal analysis shows that the profit value of increased production depends on the current level of production. Thus a savvy manager would increase production if the current level were at 450 units, but not if the current level were at 500 units.

Finding Critical Points

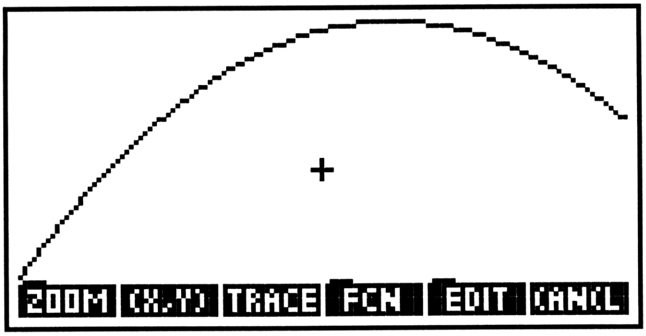
Marginal analysis raises an important point: Is there an efficient way to find the *optimum* point in a function? In the previous examples, for instance, the idea would be to find the level of production (x)—the number of units—that produces the maximum profit (y). You know that at a production level of 450 units, the marginal profit is \$80, but at 500 units the marginal profit is -\$20. So it seems likely that somewhere between 450 and 500 units, there is a level of production where the marginal profit is \$0.

Such a level would be a candidate for the optimum production level, because at that point, manufacturing either one more unit or one fewer unit yields less profit. That is, if x units is optimum, then x-1 units would have a positive marginal profit, indicating that you can improve by adding one unit, bringing the total back to x. On the other hand, if you manufacture x+1 units, you'd have a negative marginal profit, indicating that you can improve by subtracting one unit, bringing the total back to x. The *maximum* profit occurs at the point in the profit curve where the slope changes from positive to negative—i.e. *where the slope is zero.*

The HP 48 can quickly compute the *critical points* of a function, $f(x)$, which include its *roots* (points where the $f(x) = 0$), its local *maxima* and *minima* (points where $f'(x) = 0$), and its points of *inflection* (points where $f''(x) = 0$).

Example: Plot the profit function, $P(x) = -x^2 + 980x - 3000$, and then find its maximum.

1. Open the PLOT application, set the **TYPE:** to **Function**, and reset the plot parameters: [→][PLOT][▲][α][F][▼][DEL][▼][ENTER].
2. Enter the profit function into the **EQ:** field: ['][+/−][α][←][X][y^x][2][+][9][8][0][×][α][←][X][−][3][0][0][0][ENTER].
3. Set **INDEP:** to x (lower-case) and **H-VIEW** to **450 515** so as to focus on the range of x where we are seeking to find the maximum: [α][←][X][ENTER][4][5][0][ENTER][5][1][5][ENTER].
4. Check **AUTOSCALE** to let the HP 48 determine the appropriate vertical scale, and then plot the function: **✓CHK** **ERASE** **DRAW**.



5. The area of the maximum is clearly visible in the plot. To have the HP 48 compute the maximum, move the cursor near the apparent maximum and select EXTRM (Extremum) from the FCN menu: ▶ ▶ ▶ FCN EXTR. Result: EXTRM: (490,237100)

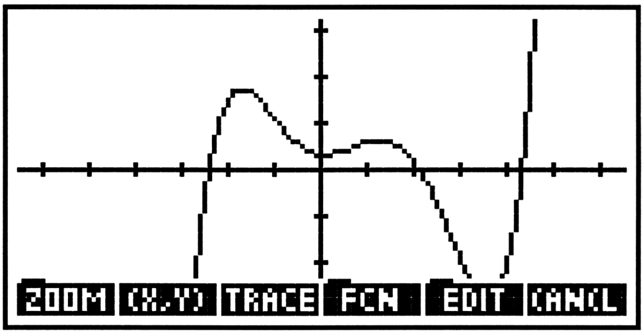
   So the maximum profit is $237,100, earned by building 490 stereos.

The profit function you just plotted had only a maximum. But there are functions that have both maxima and minima (i.e. both "humps" and "valleys"). Because the slope for both maxima and minima is zero, you don't necessarily know if the extremum found by the HP 48 corresponds to a maximum or a minimum, unless you do one of two things:

1. Narrow the search to an interval that contains the point of interest.
2. Compute the *second derivative* (i.e. the "slope of the slope") at the extremum. If the result is negative, the slope is decreasing (going from positive to negative), so the extremum is a maximum; if the result is positive, the slope is increasing (going from negative to positive), so the extremum is a minimum.

Example: Plot $x^5 - 4x^4 - 6x^3 + 20x^2 - 3x + 9$. Then narrow the search interval and use the FCN menu to find all maxima and minima.

1. Cancel the current plot, returning to the PLOT setup screen: [CANCEL].
2. Highlight the EQ: field and enter the function: [▲][▲]['][α][←][X][y^x][5][−][4][×][α][←][X][y^x][4][−][6][×][α][←][X][y^x][3][+][2][0][×][α][←][X][y^x][2][−][3][×][α][←][X][+][9][ENTER].
3. Reset H-VIEW to defaults and set V-VIEW to -100 100: [▶][DEL][ENTER][▶][DEL][ENTER][▼][◀][1][0][0][+/−][ENTER][1][0][0][ENTER].
4. Plot the function: ERASE DRAW.



5. This plot shows two local maxima ("humps") and two local minima ("valleys"). Press TRACE to trace the cursor along the plot. Then move the cursor to a spot near to the left-most maximum, and press FCN EXTR. Result (to 3 places): EXTRM: (-1.719,53.797)
6. Press [NXT] to restore the menu, move the cursor to the right until it's over the next extremum (this time, a minimum) and press EXTR.

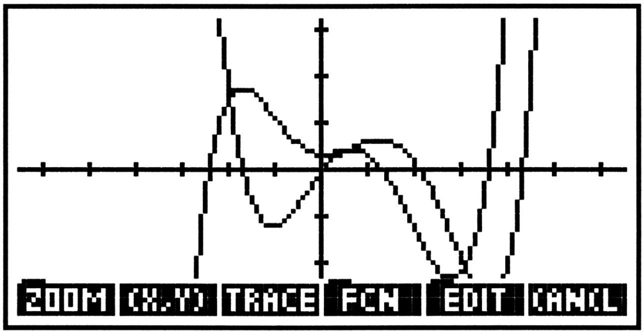
   Result (to 3 places): EXTRM: (0.078,8.885)
7. Repeat step 6 twice more to find the other maximum and minimum.

   Results (to 3 places): EXTRM: (1.246,18.068) (maximum)
   EXTRM: (3.595,-89.723) (minimum)

Perhaps a more unambiguous approach to determining maximums and minimums is to plot the first derivative function. Every *root* of the derivative function —i.e. a value where the first derivative is zero—corresponds to a maximum or minimum value in the original function.

Example: Plot the derivative over the function (currently displayed from the previous example) and find the roots of the derivative function.

1. Press NXT to redisplay the menu, then NXT F' to add the plot of the derivative function to the current function.



2. Move the cursor out near the left-most root of the derivative function and press FCN ROOT. Result (to 3 places): ROOT: -1.719
3. Move the cursor to the next root of the derivative function and press NXT ROOT. Result (to 3 places): ROOT: 0.078
4. Repeat step 3 for the other two roots:

   Results (to 3 places): ROOT: 1.246
   ROOT: 3.595

Note that the roots of the derivative function have the same independent variable value as the extrema of the original function. Furthermore, wherever the root occurs when the derivative function has a negative slope corresponds to a maximum in the original function and wherever the root occurs when the derivative function has a positive slope correspond to a minimum in the original function.

Notice that the derivative plot also has "humps" and "valleys"—local maxima and minima. These represent *points of inflection*—points where the slope of the original function switches from increasing to decreasing or vice versa. Although there is no direct computation of points of inflection on the HP 48, you can do it easily by finding the maxima and minima of the derivative of the function.

Example: Use the plot of the first derivative to find the inflection points of the original function.

1. Move the cursor near the left-most minimum on the plot of the derivative and press [NXT] **EXTR**.

   Result (to 3 places): `EXTRM: (-1.063,-40.257)`

2. Move the cursor near to the local maximum of the derivative plot and press [NXT] **EXTR**.

   Result (to 3 places): `EXTRM: (0.675,11.916)`

3. Move the cursor near the right-most minimum of the derivative plot and press [NXT] **EXTR**.

   Result (to 3 places): `EXTRM: (2.788,-76.035)`

Thus the original function, $x^5 - 4x^4 - 6x^3 + 20x^2 - 3x + 9$, has local maxima at x = -1.719 and x = 1.249, local minima at x = 0.078 and x =3.595, and points of inflection at x = -1.063, x = 0.675, and x = 2.788.

To sum up what you know about finding critical points using the FCN menu in the Picture mode on the HP 48:

Roots: Move the cursor near the desired root in the main function plot and press **ROOT**.

Extrema: Either move the cursor near the desired extremum in the main function plot and press **EXTR**; or move the cursor near the desired root of the first derivative plot and press **ROOT**.

Inflection Points: Either move the cursor near the desired extremum in the first derivative plot and press **EXTR**; or move the cursor near the desired root of the second derivative plot and press **ROOT**.

Solving Optimization Problems

The examples in this section illustrate the use of critical points in solving real-world problems that require you to find an optimum value. You've already seen one such problem when you computed the optimum production level of stereos to produce the maximum profit. Here are other problems for further practice.

The basic strategy for solving all of these problems can be summarized as follows:

1. Identify the dependent variable—the quantity you are trying to maximize or minimize.
2. Identify the constraints—the boundaries (i.e. intervals) or restrictions on the process described by the function. For example, for situations involving physical objects, a constraint is that the number of objects be nonnegative, even though mathematically a negative number may yield an optimum result.
3. Express the dependent variable as a function of a single independent variable—just as profit was expressed as a function of the number of stereos produced in the earlier example.
4. Plot the function for the range of all acceptable values of the independent variable.
5. Find the values of the function that represent the absolute maximum and/ or minimum within the range of acceptable values.
6. Use the results to answer the particular questions posed by the problem.

To illustrate this strategy, here's a good starting problem.

Problem 1: A special cylindrical packing container with closed bottom and top is to be made from two kinds of material. The material used to make the bottom and top costs \$.011 per square inch; the material for the curved outer surface of the container costs \$.006 per square inch.

The total cost of the special container is \$31.00. If r is its radius and h is its height, find the value of r that maximizes the volume.

1. Identify the dependent variable—the variable to be maximized. In this case, it is the volume, V.

2. Identify the constraints. Obviously all variables must be nonnegative since you can't have a negative radius, negative height, or negative volume. The other constraint is the total cost of the container materials. The top and bottom comprise $2\pi r^2$ square inches of surface area, and the curved sides of the container comprise $2\pi rh$ square inches. Thus the total cost of the container is:

$$.011(2\pi r^2)+.006(2\pi rh)=31$$

3. Express the volume as a function of the radius (r). The volume of a cylinder is $V=\pi r^2 h$, but this formula involves *both* radius and height. To convert it to a function of just the radius, you must express the height as a function of the radius and substitute this expression for h in the volume formula. The constraints usually determine that expression; the constraint expressed in step 2 above relates the variables of r and h, although you must still solve the expression for h:

$$h=\frac{31000-22\pi r^2}{12\pi r}$$

Use this expression for h in the volume formula and simplify:

$$V=\frac{31000}{12}r-22\pi r^3$$

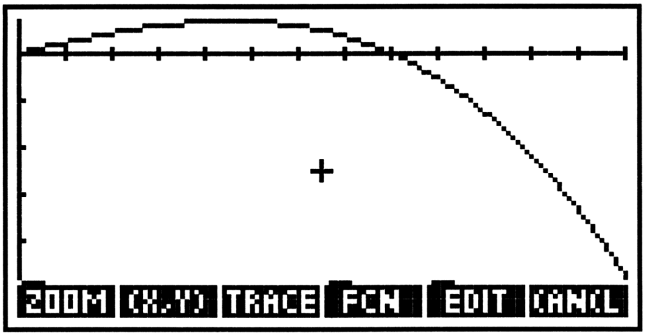
4. Plot the function for the range of acceptable values of r. Note that r must be greater than zero but can't be so large that $V<0$. Just by estimating you can see that when $r=10$, $V<0$. So a range of 0 to 10 should include the maximum volume.

   a. Open the PLOT application, and make sure that the **TYPE:** is Function. Then reset the plot: [→][PLOT][DEL][▼][ENTER].

   b. Enter the volume expression (the right-side only): ['][3][1][0][0][0] [÷][1][2][×][α][←][R][−][2][2][×][←][π][×][α][←][R][y^x][3][ENTER].

   c. Enter r for the **INDEP** variable and 0 10 for **H-VIEW**: [α][←][R] [ENTER][0][ENTER][1][0][ENTER].

   d. Check **AUTOSCALE** and draw the plot: [✓CHK][ERASE][DRAW].



5. Now find the maximum of the volume function: FCN EXTR.

   Result (to 3 places): EXTRM: (3.530, 6079.005)

6. Finally, answer the question and interpret the results. The maximum volume ≈ 6,079 cubic inches and occurs when the radius is ≈ 3.53 inches. At that volume, the height of the container is ≈ 155.29 inches. This special container (for glass rods) is nearly 13 feet long and only 7 inches wide!

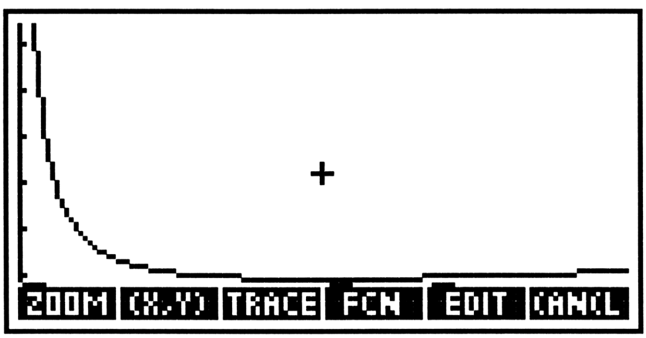
That's the general approach to optimizing. Try a second example.

Problem 2: A manufacturing company uses a particular chemical at a steady rate of 1200 gallons per year. Any number of gallons can be ordered at a time, but there is a fixed handling charge of \$100 per order, no matter the size. Storing the chemical costs the company about \$1 per gallon per year, but it must be reordered whenever the stock on hand gets down to 200 gallons. How many gallons of the chemical should be ordered each time to minimize the handling and storage charges?

1. Identify the dependent variable. You are trying to minimize the total handling and storage charges, C.
2. Identify the constraints. If n is the number of orders per year and g is the number of gallons per order, then $ng = 1200$ is one constraint.
3. Express C as a function of g, the amount ordered each time. The easiest expression for C is: $100n + s$, where s is the average number of gallons in storage. However, you need C to be expressed as a

function of g, not of n and s. Using the constraint, you can express n as a function of g: $n = 1200/g$. The *average* amount in storage, s, throughout the year is the minimum 200 gallons plus half of one order's worth. Note that the actual amount in storage at any one point is likely to be either more or less than this, but the average amount, s, is $g/2 + 200$ gallons. Substituting the expressions for n and s into the original expression for C yields $C = \dfrac{120000}{g} + \dfrac{g}{2} + 200$.

4. Plot the C function between $g = 0$ and $g = 1000$ (because 1000 gallons is the largest possible order if the constraint is to be met).
   a. Open the PLOT application, check to be sure that the TYPE: is Function, and reset the plot: [→][PLOT][DEL][▼][ENTER].
   b. Enter the expression for the volume (right-side only): ['][1][2][0][0][0][0][÷][α][←][G][+][α][←][G][÷][2][+][2][0][0][ENTER].
   c. Enter g for INDEP and 0 1000 for H-VIEW: [α][←][G][ENTER][0][ENTER][1][0][0][0][ENTER].
   d. Check AUTOSCALE and draw the plot: [✓CHK][ERASE][DRAW].



5. Compute the minimum of the function: [FCN][EXTR].

   Result (to 3 places): EXTRM: (489.898,689.898)

6. So the best order amount is 490 gallons, which will result in storage and handling charges of \$689.90 per year.

If you have a good idea of the shape of function you're optimizing and the interval containing the optimum value, you may wish to use the program MAXMIN (see page 302) instead of using the PLOT application, since it will obtain the answer more quickly.

MAXMIN takes from level 4 the function expression you would normally plot, the independent variable from level 3, a number from level 2 indicating if you're seeking the minimum (use a negative number) or maximum (use nonnegative number), and a list containing the search interval endpoints from level 1. To restrict your search to integer values (eg. stereos produced, units sold), use –1 (for minimization) or 1 (for maximization) on level 2.

Example: Use MAXMIN to solve the previous problem.

1. The first three steps of the solving process are identical whether you use MAXMIN or the plotting method. Press [CANCEL] until the stack is displayed.
2. Enter the expression: ['] [1] [2] [0] [0] [0] [0] [÷] [α] [←] [G] [+] [α] [←] [G] [÷] [2] [+] [2] [0] [0] [ENTER].
3. Enter the independent variable: ['] [α] [←] [G] [ENTER].
4. Enter a -1 to search for a minimum of the function: [1] [+/–] [ENTER].
5. Enter a list of the search interval endpoints: [←] [{ }] [0] [SPC] [1] [0] [0] [0] [ENTER].
6. Use MAXMIN to find the minimum value of *g*: [α] [α] [M] [A] [X] [M] [I] [N] [ENTER] or [VAR] (then [NXT] or [←] [PREV] as needed) [MAXM].

   <u>Result</u>:

```
      g: 490
f(g): 689.897959184
```

Problem 3: A stereo manufacturing plant has a production capacity of 25 stereos per week. Experience has shown that n articles per week can be sold at a price of p dollars each, where $p = 110 - 2n$, and the cost of producing n articles is $n^2 + 10n + 600$ dollars. How many articles should be made each week to give the largest profit?

1. The dependent variable is the profit, Q.
2. The constraints are that $n \le 25$.
3. The profit is the revenue minus the cost of producing n articles per week. Revenue of n articles is np, or $n(110 - 2n)$. Cost of n articles is $n^2 + 10n + 600$. Therefore, the profit as a function of n is:
$$110n - 2n^2 - \left(n^2 + 10n + 600\right) = -3n^2 + 100n - 600$$
4. Using MAXMIN, enter the profit function (right-hand side) and the independent variable: ['] [+/-] [3] [×] [α] [←] [N] [y^x] [2] [+] [1] [0] [0] [×] [α] [←] [N] [−] [6] [0] [0] [ENTER] ['] [α] [←] [N] [ENTER].
5. Enter a 1 to signal that you want to find the integral value of n that maximizes the profit function: [1] [ENTER].
6. Enter the endpoints of the search interval and use MAXMIN: [←] [{ }] [0] [SPC] [2] [5] [ENTER] [MAXM].

<u>Result</u>: n: 17
f(n): 233

Maximum profit ($233) is achieved by making 17 stereos per week.

Problem 4: The cost of erecting an office building is $1,000,000 for the first floor, $1,100,000 for the second, $1,200,000 for the third, and so on. Other expenses in the project (lot, basement, etc.) are $5,000,000. Assuming that the completed building will generate a net annual income of $200,000 per floor. How many floors will provide the greatest rate of return on investment? (Note: the rate of return is the revenues generated per unit of investment).

1. The dependent variable is the rate of return, I.
2. There are no special constraints other than the number of floors must be one or greater.

3. Express the rate of return, I, in terms of the number of floors built, n. At first pass, the rate of return is

$$I = \frac{\text{Revenue}}{\text{Fixed Costs} + \text{Cost per Floor}} = \frac{200000n}{5000000 + (1000000n + f(n))}$$

The $f(n)$ term reflects the extra incremental cost of each floor above the first. The increment is \$100,000 for the second floor and an additional \$100,000 for each floor after that. Looking at the increment as a cumulative sequence, beginning with the 1st floor, it is:

{ 0, 100000, 300000, 600000, 1000000, 1500000, ... }.

Express this sequence as a function of n: $100000\dfrac{n^2 - n}{2}$

(Note that the sum of the first n positive integers is the average of n and n–1 or $n(n–1)/2$.) Substitute this function into the rate of return function and simplify:

$$I = \frac{200000n}{50000n^2 + 950000n + 5000000}$$

4. Using `MAXMIN`, enter the rate of return function and the independent variable: ['] [2] [0] [0] [0] [0] [0] [×] [α] [←] [N] [÷] [←] [( )] [5] [0] [0] [0] [0] [×] [α] [←] [N] [y^x] [2] [+] [9] [5] [0] [0] [0] [0] [×] [α] [←] [N] [+] [5] [0] [0] [0] [0] [0] [0] [ENTER] ['] [α] [←] [N] [ENTER].

5. Enter a 1 to search for the integral value of n that maximizes the net rate of return and a list of endpoints for the search interval: [1] [ENTER] [←] [{ }] [1] [SPC] [2] [0] [ENTER].

6. Execute `MAXMIN` to compute the solution: [MAXM].

Result:
```
2:                n: 10
1: f(n): .102564102564
```

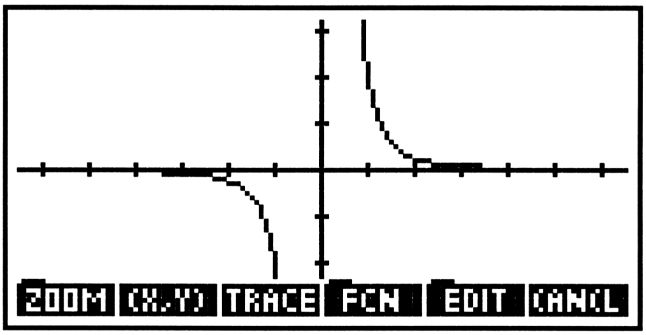
The best rate of return (≈10.26%) results from building 10 floors.

Tangents and Normals

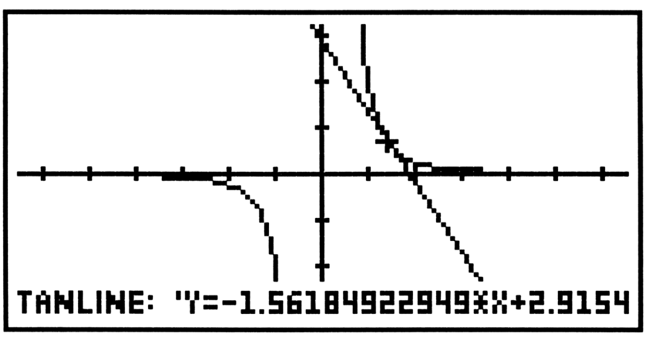
The derivative of a function at a point is the instantaneous slope of the curve at that point. The line with the same slope that contains that point on the function is the *tangent of the curve at the point*. (And the line perpendicular to the tangent line there is the *normal of the curve at the point*.)

Example: Plot $f(x) = \frac{2}{x^3}$, then compute and plot its tangent line at $x = 1.4$.

1. Open the PLOT application, check that TYPE: is Function and then reset the plot parameters: [→] [PLOT] [DEL] [▼] [ENTER].
2. In EQ:, enter the curve's expression: ['] [2] [÷] [α] [←] [X] [y^x] [3] [ENTER].
3. Enter the independent variable; plot: [α] [←] [X] [ENTER] ERASE DRAW.



4. Press TRACE (X,Y), then press [▶] until X: 1.4 is displayed.
5. Find and draw the tangent at that point: [NXT] FCN [NXT] TANL.



6. [CANCEL] [CANCEL] to the stack. The tangent equation is on level 1.

There is no built-in means of either computing or plotting the normal to the curve. However, remember that the slope of the normal is the *negative reciprocal* of the slope of the tangent (and the point of tangency is on both lines). So, with a slope m and a point (r,s) you can compute the line: $y = m(x - r) + s$.

The program TNFCN (page 330) computes both the line tangent and the line normal to a given function at a given point. It doesn't require you to plot the function first, nor does it plot anything itself. TNFCN takes the function from level 3, the independent variable from level 2, and the value of the independent variable for the point of tangency from level 1 and returns the equation of the normal to level 2 and the equation of the tangent to level 1.

Example: Use TNFCN to find the normal and tangent to $f(x) = \frac{2}{x^3}$ at $x = -2$.

1. Enter the function: ['] [2] [÷] [α] [←] [X] [y^x] [3] [ENTER].
2. Enter the independent variable and then the value of the independent variable at the point of tangency: ['] [α] [←] [X] [ENTER] [2] [+/–] [ENTER].
3. Use TNFCN to compute the tangent and normal lines to the curve at that point: [α] [α] [T] [N] [F] [C] [N] [ENTER] or [VAR] (then [NXT] or [←] [PREV] as needed) [TNFC].

Result (to 3 places):
```
2: Norm: 'y=2.667*x+5.083'
1: Tang: 'y=-(.375*x)-1'
```

Example: Find the normal and tangent to $f(x) = \cos x$ at $x = 0$.

1. Enter the function: ['] [COS] [α] [←] [X] [ENTER].
2. Enter the independent variable and then the value of the independent variable at the point of tangency: ['] [α] [←] [X] [ENTER] [0] [ENTER].
3. Use TNFCN to compute the tangent and normal lines to the curve at the given point: [α] [α] [T] [N] [F] [C] [N] [ENTER] or [VAR] ([NXT] or [←] [PREV] as needed) [TNFC].

Result :
```
2: Norm: 'x=0'
1: Tang: 'y=1'
```

Here the tangent is a horizontal line and the normal is a vertical line.

Finding the Angle Between Two Curves at a Point

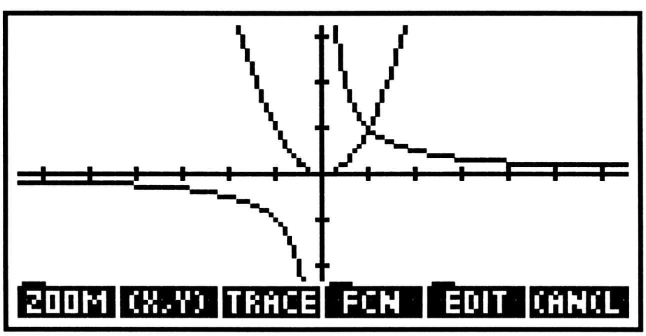
The angle two curves make when they intersect is the angle between their *tangents* at the point of intersection. If the slopes of the two curves are m_1 and m_2, then the angle of intersection is: $\theta = \tan^{-1}\left(\dfrac{m_1 - m_2}{1 + m_1 m_2}\right)$. The angle computed by this formula is the angle from the first tangent to the second in the clockwise direction.* If you're given two curves, you must do the following to find the angle they form at their intersection:

1. Determine the coordinates of the point of intersection.
2. Determine the slopes of the each of the curves at the point of intersection.
3. Compute the angle using the formula above.

Most of the difficulty comes in determining the point of intersection.

Example: Find the angle between $y = x^2$ and $y = 1/x$ where they intersect.

1. For two curves that are both functions (i.e. they both meet the vertical line test), you can find their intersection(s) by plotting them. Open PLOT, check that `TYPE:` is set to `Function`, and reset the plot parameters: [→][PLOT][DEL][▼][ENTER].
2. In the `EQ:` field enter a list of the equations of the two curves: [←][{ }] ['][α][←][X][y^x][2][▶]['][1][÷][α][←][X][ENTER].
3. Set the independent variable to `X` and plot the curves: ['][α][←][X] [ENTER] **ERASE** **DRAW**.



*Of course, the two curves form a second angle when they intersect—the supplement of the one you computed.

4. Find the point of intersection: FCN ISECT.

   Result: I-SECT: (1,1)

5. Return to the stack and store the coordinates of the point of intersection in x and y respectively: [CANCEL] [CANCEL] [MTH] [NXT] CMPL C→R ['] [α] [←] [Y] [STO] ['] [α] [←] [X] [STO].

6. Compute the slope of the first function at (1,1): ['] [α] [←] [X] [y^x] [2] [ENTER] ['] [α] [←] [X] [ENTER] [→] [∂]. Result: 2

7. Compute the slope of the second function at (1,1): ['] [1] [÷] [α] [←] [X] [ENTER] ['] [α] [←] [X] [ENTER] [→] [∂]. Result: -1

8. Using the two slopes, find the angle of intersection (in Deg mode): ([←] [RAD], if needed to change to Deg mode) [−] [1] [ENTER] [2] [ENTER] [1] [+/−] [×] [+] [÷] [←] [ATAN] [MTH] VECTR ABS.

   Result (to 3 places): 71.565

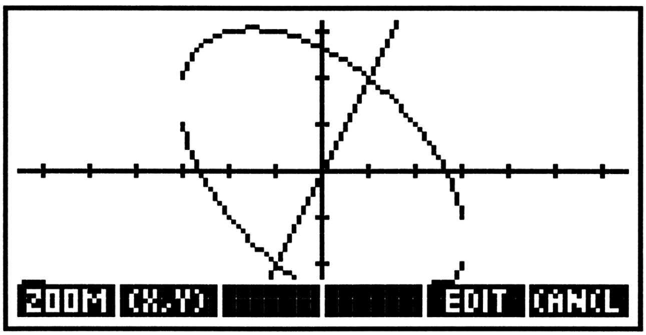
The previous example is a very simple one because it involves only curves with explicitly separated independent and dependent variables whose points of intersection are easy to determine. The next few examples illustrate how to deal with other, more complicated sets of curves.

If one or more of the curves are expressed so that you cannot easily separate independent and dependent variables, you must use implicit differentiation to compute the slope of its tangent. Furthermore, you'll need to use either the Conic plot type (for curves of second degree or less) or the SOLVPLT program (for curves of third degree and higher—see page 325) to view their intersections. But then that brings up an additional problem: how can you determine the point or points of intersection? The Conic plot type doesn't have the FCN menu commands (such as ISECT) available to it.

If you can separate the independent and dependent variables in one of the curves then you can find the points of intersection by substitution. The following example shows you an illustration of this.

Example: Plot the following curves and then find the angle between them at their points of intersection: $x^2 + xy + y^2 = 7$; $y = 2x$.

1. Open PLOT, change the `TYPE:` to `Conic`, and reset the plot parameters: [→][PLOT][▲][α][C][▼][DEL][▼][ENTER].
2. In `EQ:`, enter the curves as a list: [←][{ }]['][α][←][X][y^x][2][+][α][←][X][X][α][←][Y][+][α][←][Y][y^x][2][←][=][7][▶]['][α][←][Y][←][=][2][X][α][←][X][ENTER].
3. Change `INDEP:` to `x` (lower-case) and `DEPND:` to `y`, then draw the plot using the default viewing ranges: [α][←][X][ENTER] [OPTS] [▼][▶][α][←][Y][ENTER][ENTER] [ERASE] [DRAW].



As you can see, there are two intersection points.

4. Although the Conic plot type has no FCN menu to compute ISECT, you can move the cursor near the points of intersection and view the approximate coordinates using [(X,Y)]. You see that the points of intersection are approximately (-1,-2) and (1,2). Cancel the plot and redisplay the stack: [CANCEL][CANCEL].
5. Now compute the exact intersection of the curves. Because the second curve (actually, it's a line) is essentially a definition of y in terms of x, you can substitute the definition for y in the first curve. To do this, make sure that x is a formal variable, then store `'2*x'` in the variable `y`, and enter the first equation: ['][α][←][X][ENTER][←][PURG]['][2][X][α][←][X][ENTER]['][α][←][Y][STO]['][α][←][X][y^x][2][+][α][←][X][X][α][←][Y][+][α][←][Y][y^x][2][←][=][7][ENTER].
6. Substitute the expression for y by evaluating the expression for the first curve: [EVAL][←][SYMBOLIC] [EXPA] [COLCT]. Result: `'7*x^2=7'`

7. Solve the resulting equation for x. In this case you can see that the equation resolves to $x^2=1$ so that $x = \pm 1$. (You can use the Solve application to solve for x in more complicated cases.) At $x = 1$, $y = 2$ and at $x = -1$, $y = -2$, as you can see by using the second curve equation. Thus the points of intersection were exactly (-1,-2) and (1,2) not merely the estimate you obtained from the plot in step 4!

8. Now compute the slopes for the two curves at their points of intersection. The slope for the second curve is constant: 2. To find the slope for the first curve, use implicit differentiation (see page 72). Purge y again, enter the first curve, then use the program IMPδ: ['] [α][←][Y] [←][PURG] ['][α][←][X] [y^x] [2] [+] [α][←][X] [×] [α][←][Y] [+] [α][←][Y][y^x][2][←][=][7][ENTER][←][{ }][α][←][X][SPC][α][←][Y][ENTER] [VAR] (then [NXT] or [←][PREV] as needed) **IMPδ**.

   Result: '2*x+(y+x*δy.x)+δy.x*2*y=0'

9. The 'δy.x' term means "derivative of y with respect to x"—i.e., the slope. Now, solve for the slope at two intersection points with the Solve application. Open the Solve application ([→][SOLVE][ENTER]) and install the implicit derivative expression on level 1 of the stack as the current equation in EQ: [NXT] **CALC** ([DROP], if necessary to bring the derivative to level 1) **OK**.

10. Enter the set of values for X and Y corresponding to the first point of intersection and solve for δY.X: [▼][1][ENTER][2][ENTER] **SOLVE**.

    Result: δY.X: -.8

11. Enter the set of values for X and Y corresponding to the second point of intersection and solve for δY.X: [▲][▲] [1][+/–] [ENTER] [2][+/–] [ENTER] **SOLVE**. Result: δY.X: -.8

    The same result as before: the tangent lines at the two points of intersection are parallel.

12. Find the angle of intersection at both points of intersection: [CANCEL][2] [–] [1] [ENTER] [•][8][+/–] [ENTER] [2][×] [+] [÷] [←] [ATAN] [α][α][A][B][S] [ENTER]. Result: 77.905

    Note that you need to make just one computation because the slopes of the two curves are identical at both points.

The program C1θC2 (see page 287) automates the computation of the angle once you know the point or points of intersection. It takes a list of curves from level 3, a list of the variables from level 2, and the point(s) of intersection as a list of order pairs of coordinates from level 1. The coordinates in level 1 should be in the same order within the ordered pair as their corresponding variable names are listed on level 2. The angle of intersection—displayed according to the current angle mode—is returned to level 1.

Example: Repeat the previous example using C1θC2. Assume that you already know the points of intersection.

1. Enter a list of the curves. Remember: both expressions should have two variables: [←][{ }][']][α][←][X][y^x][2][+][α][←][X][×][α][←][Y][+][α][←][Y][y^x][2][←][=][7][▶][']][α][←][Y][←][=][2][×][α][←][X][ENTER].
2. Enter a list of the variables: [←][{ }][α][←][X][SPC][α][←][Y][ENTER].
3. Enter a list of the points of intersection, each point expressed as an ordered pair of coordinates: [←][{ }][←][( )][1][←][,][2][▶][←][( )][1][+/–][←][,][2][+/–][ENTER].
4. Compute the angles between the two curves at the points of intersection using C1θC2: [α][α][C][1][→][F][C][2][ENTER] or [VAR] (then [NXT] or [←][PREV] as needed) [C1θC2].

   Result (to 3 places): { :θ(1,2): 77.905
   :θ(-1,-2): 77.905 }

   Note that this result assumes Deg mode like the previous example. If you were in Rad mode, the answer would be 1.360 radians.

Now for the final complication.... The points of intersection of the previous set of curves were easy to find because it was possible to make a substitution of one variable with another. How do you find the point of intersection in situations when it isn't easy to perform such a substitution?

Two special tools need to be added to those that come built into your HP 48. SOLVPLT is a program that will plot any two-variable relation or set of two-variable relations, by combining the Solver and Function plotting capabilities of the HP 48. NLSYS solves a system of non-linear equations starting from an initial estimate of the solution. Since non-linear systems may have more than one solution, NLSYS can find different solutions depending upon the initial estimate you give it—much as the built-in Solve application does when solving for a missing variable. SOLVPLT is useful for determining the number of points of intersection and for giving an estimate of their coordinates, while NLSYS can take the estimate coordinates and compute the exact (within limits of machine precision) solutions.

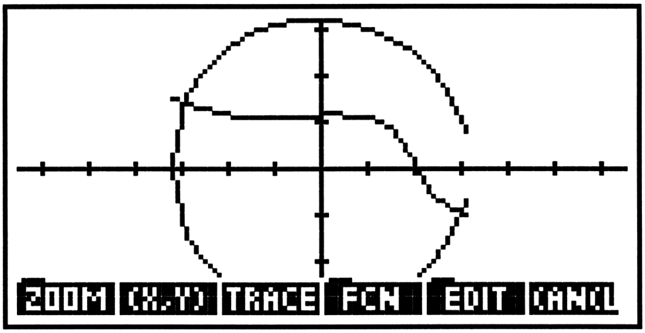
SOLVPLT (see page 325) takes a list of the curves from level 5, a list of independent and dependent variables (independent listed first) from level 4, a list containing the low and high endpoints of the plotting range from level 3, a list of starting estimates of the dependent variable for each curve from level 2, and a positive integer representing the resolution of the plot from level 1. Larger level 1 integers lead to fast-and-rough plots, while smaller level 1 integers give slower-but-more-precise plots. The list of starting estimates on level 2 should have one entry for each curve. However, an "entry" can itself be a list of two or more values, if its corresponding curve is one with more than one branch—such as a conic. This allows SOLVPLT to draw all branches of complicated curves if you choose starting estimates wisely.

NLSYS (see page 304) takes a list of curves from level 3, a list of the variables from level 2, and a set of starting guesses for each of the variables—either as a list or as a complex number (ordered pair)—on level 1. All of the lists must contain the same number of elements. The curves listed on level 3 must each be expressions equal to zero. For example, the polynomial x^3+4x^2-3x+5 should be expressed as $x^3+4x^2-3x+5-y$, reflecting that the polynomial is equal to y by implication. There should always be a minimum of two curves and two variables when using NLSYS.

The following example illustrates the use of both SOLVPLT and NLSYS:

Example: Find the angle between $x^3+x^2y-xy^2+6y^3=2$ and $x^2+y^2=4$ where they intersect.

1. Enter the set of curves as expressions equal to zero: [←][{ }]['][α][←][X] [y^x][3][+][α][←][X][y^x][2][×][α][←][Y][−][α][←][X][×][α][←][Y][y^x][2][+] [6][×][α][←][Y][y^x][3][−][2][▶]['][α][←][X][y^x][2][+][α][←][Y][y^x][2][−] [4][ENTER].
2. Enter the set of variables: [←][{ }][α][←][X][SPC][α][←][Y][ENTER].
3. Find the curves' points of intersection. Because NLSYS requires you to enter a reasonable guess to seed the search for points of intersection, it behooves you to plot the curves—using SOLVPLT—to deter-mine the number and approximate location of all intersection points.
   a. Enter the list of curves and the list of variables—which you can do simply by making a copy of the first two stack levels. Make a second set of copies for later: [←][STACK][NXT] DUP2 DUP2.
   b. Enter the plotting range. The second curve is a circle of radius 2, so a range of -2 to 2 is reasonable: [←][{ }][2][+/−][SPC][2][ENTER]
   c. Enter a list of starting estimates for y for each of the curves. Since the first value of x is -2 for the given plotting range, estimate the value of y when x = -2 for each of the curves. For the first curve, an estimate of about 1 seems close. For the second curve, a circle, you need a list of starting points, so that both "halves" of the circle are drawn (use -.001 and .001 for example). Enter the estimates as a list, `{ 1 { -.001 .001 } }`: [←][{ }][1][SPC] [←][{ }][•][0][0][1][+/−][SPC][•][0][0][1][ENTER].
   d. Enter 4 to set the plot speed/resolution: [4][ENTER].
   e. Use SOLVPLT to plot the curves: [α][α][S][O][L][V][P][L][T][ENTER] or [VAR] (then [NXT] or [←][PREV] as needed) SOLVP....



f. There appear to be two points of intersection. Move the cursor near each point of intersection and press [ENTER] to copy the coordinates to the stack, to be used as the estimates NLSYS. Press [CANCEL] when you've finished.

   Result (to 3 places):

   ```
   2:        (-1.785,0.857)
   1:        (1.846,-0.603)
   ```

g. Gather the estimates into a list, then make copies of the curves and variables and gather each into a two-element list: [2] [PRG] LIST →LIST [3] [←] [STACK] ROLLD [ENTER] [2] [PRG] LIST →LIST [3] [←] [STACK] ROLLD [ENTER] [2] [PRG] LIST →LIST [3] [←] [STACK] ROLLD.

h. Use NLSYS (in list processing style) to compute the actual points of intersection: [3] [ENTER] [←] [« »] [α] [α] [N] [L] [S] [Y] [S] [ENTER] [PRG] LIST PROC DOLIS.

   Result (to 3 places):

   ```
   { { :x: -1.807 :y: 0.857
     { :x: 1.892 :y: -0.650 } }
   ```

4. Rearrange the previous result so that it's in the proper form and compute the angles at the given points using C1θC2: [1] [ENTER] [←] [« »] [EVAL] [MTH] [NXT] CMPL R→C [ENTER] [PRG] LIST PROC DOLIS [VAR] (then [NXT] or [←] [PREV] as needed) C1θC2.

   Result (to 3 places and in Deg mode):

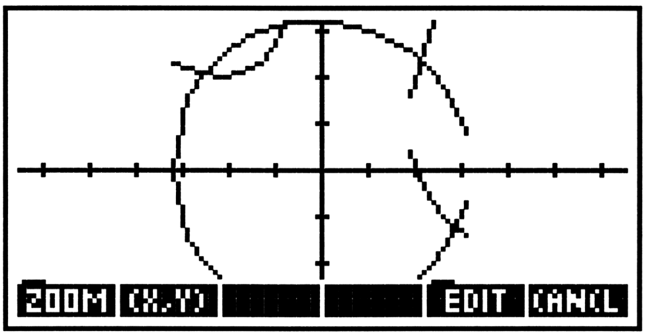
   ```
   { :θ(-1.807,0.857): 81.573
     :θ(1.892,-0.650): 79.008 }
   ```

   The two angles are ≈81.6° and ≈79.0°.

One last point needs to be mentioned.

You may have noticed that SOLVPLT takes some time to do its work (and that's plotting only one of each four pixels!). The Conic plot type will plot the second-degree Taylor's polynomial of any third-degree (or higher) relation of two variables and can also be used to determine starting estimates. For many cases, it may plot faster, although the visual results are deceiving.

For example, using the built-in Conic plot type to plot the two curves in the previous example, using the same plotting and viewing ranges, and default resolution yields:



There appear to be four points of intersection—even though the "true" plot shows only two points of intersection. The good news, however, is that if you were to use four estimates based on this plot with NLSYS, you would get the two actual points of intersection just as before—you would simply get multiple copies of one or both of them. If you're aware of the potential deception of Conic plot, it can be a speedier alternative to using SOLVPLT for higher-order relations.

Related Rates of Change

Probably the most common use for derivatives is to analyze processes that change with time—situations in which time t is the independent variable. In many common real-world contexts, processes that change with time involve several related quantities that all simultaneously change with time.

For example, as a balloon is inflated, there are a large number of geometric and physical attributes—volume, surface area, radius, diameter, circumference, weight, internal pressure, and temperature—all changing simultaneously. Since many of these variables are related to each other by mathematical formulas and physical laws, it should be possible to determine the rate of change of one attribute if you know the rate of change of a related attribute—and if you can describe that relationship mathematically.

This kind of question is often referred to as a *related rates problem.* Here's how to attack related rates problems:

1. Determine which quantities are changing (variables) and which are constant in a given problem. This may involve drawing a diagram of the problem to visualize the process involved.
2. Express a mathematical relationship between the variables. Keep it general—don't substitute measured values for any of the variables yet.
3. Differentiate the expression with respect to time. This usually requires an implicit differentiation because you must treat all variables as functions of time, the variable of differentiation. The result will be an expression relating the *rates of change* of the various quantities.
4. Substitute all known variable and rate values for the instant in time in which you're interested. Check to make sure that you're using compatible units. For example, if a distance is measured in inches but a rate is given in ft/sec., you may need to divide the distance by 12 before solving the expression.
5. Solve for the quantity or rate that you require. Be careful in this final step, too, to use compatible units.

Example: A spherical balloon of radius 3 cm is being heated. If its radius is increasing at 5 mm per minute, how fast is the volume increasing?

1. *Determine the variables and the constants.* The radius r and the volume V are the variables in this problem.

2. *Express a mathematical relationship between the variables.* The volume of a sphere is defined in terms of its radius as $V = \frac{4}{3}\pi r^3$.

   Enter the expression onto the stack: ['] [α] [V] [←] [=] [4] [÷] [3] [×] [←] [π] [×] [α] [←] [R] [yˣ] [3] [ENTER].

3. *Implicitly differentiate the expression with respect to time.* Enter a list of variables, with the time , t , first: [←] [{ }] [α] [←] [T] [SPC] [α] [V] [SPC] [α] [←] [R] [ENTER]. Then use the implicit differentiation program, IMPδ: [VAR] (then [NXT] or [←] [PREV] as needed) **IMPδ**.

   Result: 'δV.t=1.333*π*(δr.t*3*r^2)'

   Remember that, by convention, 'δV.t' is $\frac{dV}{dt}$ and 'δr.t' is $\frac{dr}{dt}$.

4. *Substitute all known values.* The two known values are the rate of change of the radius (δr.t), 5 mm/min; and the radius (r), 3 cm. At this point, you must choose: If you include units, you must give the correct units for *all* variables in the expression, even the values. So in this problem, you'd store a valid unit for δV.t, such as 0 cm³/min, besides the known values. If you don't include units, you must manually adjust values so as not to omit unit conversion factors: either change the radius rate to 0.5 cm/min or the radius to 30 mm.

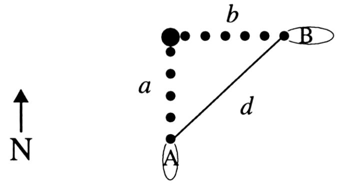
   This time, store the values with units: [5] [→] [UNITS] **LENG** **MM** [→] [UNITS] **TIME** [→] **MIN** ['] [α] [→] [D] [α] [←] [R] [·] [α] [←] [T] [STO] [3] [→] [MENU] **CM** ['] [α] [←] [R] [STO] [0] [→] [UNITS] **VOL** **CM^3** [→] [UNITS] **TIME** [→] **MIN** ['] [α] [→] [D] [α] [V] [·] [α] [←] [T] [STO].

5. *Solve for the quantity or rate desired.* Open the SOLVE EQUATION application and retrieve the differentiated expression from the stack using the CALC feature: [→] [SOLVE] [ENTER] [NXT] **CALC** ([DROP], if necessary to bring the expression to level 1) **OK**. Then make sure the δV.T: field is highlighted (it should be) and press **SOLVE**.

   Result (to 3 places): δV.T: 56.549_cm^3/min

Example: Two boats are racing at constant speed toward a finish marker, boat A from the south at 13 mph and boat B from the east. When equidistant from the marker, the boats are 16 miles apart and the distance between them is decreasing at 17 mph. Which boat will win?

1. *Determine the variables and constants.* A diagram is best:



Here *a* is the distance between boat A and the finish marker; *b* is the distance between boat B and the finish marker; and *d* is the distance between the bows of the two boats. All distances change with time.

2. *Express a mathematical relationship between the variables.* Clearly, it's the Pythagorean theorem: $a^2 + b^2 = d^2$. Enter the expression: ['] [α][←][A][y^x][2][+][α][←][B][y^x][2][←][=][α][←][D][y^x][2][ENTER].

3. *Implicitly differentiate the expression with respect to time.* Enter a list of the variables, with *t* as the first in the list, and execute `IMP∂`: [←][{ }][α][←][T][SPC][α][←][A][SPC][α][←][B][SPC][α][←][D][ENTER][VAR] **IMP∂**. Result: `'∂a.t*2*a+∂b.t*2*b=∂d.t*2*d'`

   Open the **SOLVE EQUATION** application and retrieve the expression to the **EQ:** field: [→][SOLVE][ENTER][NXT] **CALC** [DROP] **OK**.

4. *Substitute known values for the appropriate variables and rates.* `∂a.t` = -13 mph (negative since *a* is decreasing); `d` = 16 miles; `∂d.t` = -17 mph. That leaves two variables (`a` and `b`) and one rate (`∂b.t`) unknown—a situation impossible to solve. But `a` = `b` at the moment in question, so $2a^2 = 2b^2 = 16^2$. Solving for *a* (and thus also for *b*) yields ≈11.3137. Store the values in the appropriate variables. Because the units are consistent throughout (miles and mph), you need not include them: [1][3][+/−][ENTER][1][1][•][3][1][3][7][ENTER] [▶][1][1][•][3][1][3][7][ENTER][1][7][+/−][ENTER][1][6][ENTER].

5. *Solve for the desired quantity.* The missing value is the speed of boat B (`∂b.t`) at the moment of decision. If it's faster than boat A (13 mph) B will win; otherwise, boat A will win. Highlight **∂B.T:** and **SOLVE**. Result (to 1 place): **∂B.T: -11.0** Boat A wins!

The RELRT Program

The program RELRT (see page 318) assists you in solving related rates problems. It helps manage the change of variable names necessary for proper differentiation and keeps track of all the variables involved in such problems.

To begin RELRT, put the relationship function (or list of functions) on level 3 of the stack. Put a list of the variables in the problem on level 2. If time is an explicit variable in any of the relationship functions, use 't' as its name and make sure it comes first in the level 2 list of variables. On level 1, enter the name (or list of names) of the variables you're solving for. If you want to solve for a *rate of change* of a variable (i.e. its derivative with respect to time), use the convention first described on page 71. For example, to solve for the rate of change of volume, *V*, you would enter 'δV.t' (read "derivative of *V* with respect to *t*") on level 1 of the stack. The δ character is [α][→][D].

After you've loaded the three stack levels properly, launch RELRT. It will compute a few things and then prompt you to enter the values of the known variables. By convention, a zero following a variable name (eg. V0) means the value of the quantity at time = 0. A t following a variable name (eg. Vt) means the value of the quantity at the moment in time being examined.

As you enter values for the various variables, you must choose: *do you include units?* RELRT can perform unit conversions, but if you include units, you must input a unit for *every* variable and *every* rate in the problem (for values for which you have no information, you must enter a *zero* with the valid unit attached). The exceptions are the initial values—those names such as V0, ending in 0.* If you choose *not* to include units, be sure to perform manually any unit conversions of the values necessary to make them consistent with one another and enter values just for the variables you know something about, leaving the unknowns blank.

Once you've entered values (with or without units attached) for the variables, press OK and the variable(s) you included on level 1 originally will be solved for and the answer returned to level 1 of the stack.

*Not all problems require or involve status of variables at time = 0. For these kinds of problems, the initial values can be (and must be) left blank, even when using units.

Example: An empty underground storage tank has the shape of a cone (vertex down) 20 feet deep and 40 feet in diameter. If water is pumped into it at a constant rate of 100 gallons per minute, how fast is the water depth in the tank changing 10 minutes after the start of pumping?

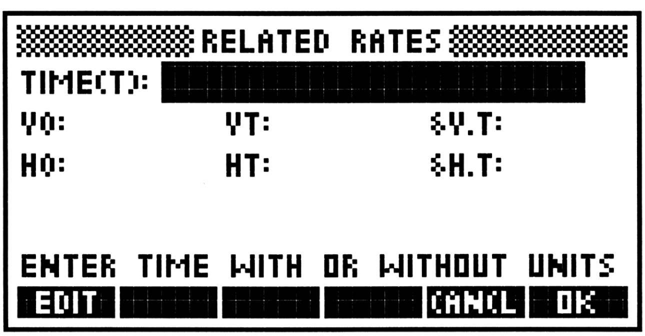
1. *Determine the variables.* A cone's volume is $V = \frac{1}{3}\pi r^2 h$, which represents the volume of water in the tank at any given point, where V is the volume of water, r the radius of the surface of the water and h the depth of the water. All three of these vary with time.

2. *Express the relationship between the variables.* You already have an expression, but it can be simplified further. Notice the relationship between r and h: If the diameter of the tank is 40 feet, then its radius r is 20 feet—equal to its depth h. Thus, you can conclude that $r = h$, and because you are asked to solve for a rate involving the depth h, the expression for the volume is reduced to $V = \frac{1}{3}\pi h^3$.

   Enter the expression: ['] [α] [V] [=] [1] [÷] [3] [×] [←] [π] [×] [α] [←] [H] [y^x] [3] [ENTER].

3. *Begin the* RELRT *program.* Enter a list of the variables in the expression: [←] [{ }] [α] [V] [SPC] [α] [←] [H] [ENTER]. Enter the variable or rate for which you are solving. In this case, it is the rate of change of the depth h, or 'ðh.t': ['] [α] [→] [D] [α] [←] [H] [•] [α] [←] [T] [ENTER].

   Finally, execute RELRT: [α] [α] [R] [E] [L] [R] [T] [ENTER] or [VAR] ([NXT] or [←] [PREV]) RELRT. After a few moments you'll see this display:



4. *Store values with units in the known variables and rates and store a zero value with the appropriate unit in the unknown variables and rates*. Known values are: $t = 10$ min; $\delta V.t = 100$ gal/min. Units for unknowns are: $V_t = 0$ gal; $h_t = 0$ ft; $\delta h.t = 0$ ft/min. Leave V_0 and h_0 blank because they aren't used in this problem (the situation at $t = 0$ isn't relevant here).

   Store the values and units: [1][0][→][_][α][←][M][α][←][I][α][←][N][ENTER][▶][0][→][_][α][←][G][α][←][A][α][←][L][ENTER][1][0][0][→][_][α][←][G][α][←][A][α][←][L][÷][α][←][M][α][←][I][α][←][N][ENTER][▶][0][→][_][α][←][F][α][←][T][ENTER][0][→][_][α][←][F][α][←][T][÷][α][←][M][α][←][I][α][←][N][ENTER].

5. *Solve for the specified variable*. Inspect the values you've entered in the previous step and when you're satisfied they're correct, press [OK]. You'll see the message `Solving . . .`, and after a bit, the solution will be returned to the stack.

   Result (to 3 places): `δh.t: 0.168_ft/min`

   After 10 minutes of pumping, the water level is rising at about two inches per minute.

Example: A weather balloon is tethered so that it stays at a constant height of 300 meters. The wind blows it horizontally at a rate of 8 m/second from its original position. If the line is spooled out so that the altitude remains constant, how fast must the line be let out when the balloon lies over a point on the ground 500 meters from the spool?

1. *Determine the variables.* Let L be the length of the unspooled line, h be the altitude of the balloon, and x be the horizontal displacement of the balloon from its original position. However, only L and x are variables; h is a constant (300 m).

2. *Express the relationship between the variables.* A right triangle is formed, with L being the hypotenuse and x and h the legs. Thus, $x^2 + h^2 = L^2$ or, substituting for h, $x^2 + 300^2 = L^2$. Enter the expression: ['][α][⬅][X][yˣ][2][+][3][0][0][yˣ][2][=][α][L][yˣ][2][ENTER].

3. *Begin the* RELRT *program.* Enter a list of the variables in the expression: [⬅][{ }][α][⬅][X][SPC][α][L][ENTER]. Enter the variable or rate for which you are solving. In this case, it's the rate of change of the unspooled line length L, or 'δL.t': ['][α][➡][D][α][L][·][α][⬅][T][ENTER]. Now execute RELRT: [α][α][R][E][L][R][T][ENTER] or [VAR] (then [NXT] or [⬅][PREV] as needed) **RELRT**.

4. *Store values without units in the known variables and rates.* Since your expression includes the value of h without units, you must be consistent now and not use units. The known values are: $x = 500$; $\delta x.t = 8$. Store just those knowns: [▼][▶][5][0][0][ENTER][8][ENTER].

5. *Solve for the specified variable.* Inspect the values you've entered in the previous step and when you're satisfied they're correct, press **OK**. You'll see the message Solving . . ., and after a bit, the solution will be returned to the stack.

Result (to 3 places): δL.t: 6.860

Because the units are not included, you must add them yourself: the line is unspooling at nearly 7 meters per second in order to maintain constant altitude in the very brisk wind.

Example: In the special theory of relativity, the mass of a particle moving at velocity v is $m\left(1-\frac{v^2}{c^2}\right)^{-\frac{1}{2}}$, where m is the mass at rest and c is the speed of light. At what rate is the mass changing when the particle's velocity is 0.5c and the rate of change of the velocity is 0.01c per second? What is the mass of a particle traveling at 0.5c whose rest mass is one unit? The speed of light, c,is 3.00E8 m/s.

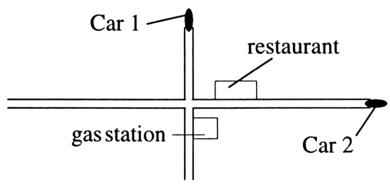
1. *Determine the variables and the constants.* The variables are the mass m and the velocity v. The speed of light, c, is a constant. In the previous example, you replaced the value of the constant directly in the main expression. This time, instead of replacing c with its value in the expression, just store its value now: [3] [EEX] [8] ['] [α] [←] [C] [STO].
2. *Express the relationship between variables.* If the rest mass is 1 unit, the mass at velocity v is $m=\left(1-\frac{v^2}{c^2}\right)^{-\frac{1}{2}}$. Enter it: ['] [α] [←] [M] [←] [=] [←] [()] [1] [−] [α] [←] [V] [yˣ] [2] [÷] [α] [←] [C] [yˣ] [2] [▶] [yˣ] [•] [5] [+/−] [ENTER].
3. *Begin the* RELRT *program.* Enter a list of the variables: [←] [{ }] [α] [←] [M] [SPC] [α] [←] [V] [ENTER]. You're solving for 2 variables here: the change in mass at time t ('δm.t'); and the mass at time t ('mt'). Enter those two variables in a list: [←] [{ }] [α] [→] [D] [α] [←] [M] [•] [α] [←] [T] [SPC] [α] [←] [M] [α] [←] [T] [ENTER]. Now use RELRT: [α] [α] [R] [E] [L] [R] [T] [ENTER] or [VAR] ([NXT] or [←] [PREV] as needed) **RELRT**.
4. *Store values without units in the known variables and rates.* Known values are: v=.5c; δ$v.t$=.01c. Store the known values: [▼] [▼] [▶] ['] [•] [5] [×] [α] [←] [C] [ENTER] ['] [•] [0] [1] [×] [α] [←] [C] [ENTER].
5. *Solve for the specified variables.* Inspect the values you just entered. When satisfied, press **OK**. You'll see `Solving . . .`, then the solution (to 4 places):

```
2:     δm.t: 0.0077
1:       mt: 1.1547
```

Its rest mass was 1 unit, so the particle is gaining mass at 0.77%/sec as it travels at half the speed of light; and it has a mass of 1.1547 units, having gained more than 15% of its rest mass.

Example: Two roads intersect at right angles. Car 1 leaves a gas station located 2 miles south of the intersection, traveling north at 30 mph. At the same time, Car 2 leaves a restaurant parking lot located 3 miles east the intersection, traveling east at 40 mph. How fast is the distance between them changing 45 minutes later?

1. *Determine the variables and the constants.* Let d be the distance between the two cars, x be the distance of car 1 north of the intersection and y the distance of car 2 east of the intersection:



2. *Express the relationship between the variables.* The relationship between x, y and d is the Pythagorean theorem: $x^2 + y^2 = d^2$. Enter this: ['][α][↰][X][y^x][2][+][α][↰][Y][y^x][2][↰][=][α][↰][D][ENTER].

3. *Begin the* RELRT *program.* Enter a list of the variables: [↰][{ }][α][↰][X][SPC][α][↰][Y][SPC][α][↰][D][ENTER]. Enter the solution variable, the rate of change of the distance d between the cars (δd.t): ['][α][↱][D][α][↰][D][•][α][↰][T][ENTER]. Finally, execute RELRT: [VAR] RELRT.

4. *Store values without units in the known variables and rates.* This problem uses starting positions x0 and y0. Note that the starting position of car 1 is *minus* 2 miles because it starts south of the intersection. Thus, known values are: $t = 0.75$; $x_0 = -2$; $y_0 = 3$; $\delta x.t = 30$; $\delta y.t = 40$. Store the known values: [•][7][5][ENTER][2][+/−][ENTER][▶][3][0][ENTER][3][ENTER][▶][4][0][ENTER].

5. *Solve for the specified variables.* Inspect the values you've entered in the previous step and when you're satisfied they're correct, press OK. You'll see Solving . . ., then the solution.

Result (to 3 places): δd.t: 49.808

So, after 45 minutes, the distance between the cars is increasing at nearly 50 mph.

Notes

4. Integration and the Integral

The Integral: Number, Function, Family of Functions

The term *integral* is used for three distinct purposes:

1. The *definite integral* is the (signed) area bounded by a function $y = f(x)$ and the x-axis and by vertical lines through the two x-value *limits of integration*.
2. The *general indefinite integral* is the family of functions whose derivative is equal to a given function.
3. The *specific indefinite integral* is the one function whose derivative is a given function that includes a specific point.

So, an *integral* of a function can be either a single number, a single function, or a family of functions, depending on the current context.

Actually, the process of determining the area "under" a curve is best described as integration while the process of finding the indefinite integral (either general or specific) is best described as *antidifferentiation*—i.e. "undoing" the process of differentiation. For this reason you may see the term *antiderivative* used interchangeably with *indefinite integral*. Perhaps *integration* and *integral* should be used when the context is area measurement and description, while *antidifferentiation* and *antiderivative* should be used when the context is finding a function or family of functions equal to a given derivative. However, this book will not be that picky and will use the two terms interchangeably.

Now, the HP 48 comes with the built-in capability to integrate functions using numeric, graphic, and symbolic techniques. Specifically:

- It can compute a numerical estimate of the definite integral of any function if it is given two finite numeric limits and if its integrand contains no undefined variables other than the variable of integration.
- It can graphically estimate the area under a plotted function between two values of the independent variable.
- It can symbolically integrate (i.e. find the antiderivative of) any polynomial function. It can also use symbolic variables in either of the limits.
- It can symbolically integrate certain other functions that are in a form that matches the set of patterns built into its memory.

Numeric Integration

The integration function on the HP 48, located on the keyboard ([→][∫]) computes a definite integral—a number. Thus, it requires four inputs:

1. *Integrand*: the function being integrated.
2. *Variable of integration*: the independent variable of the function.
3. *Lower limit*: the value of the independent variable representing the lower boundary of the integrable region.
4. *Upper limit*: the value of the independent variable representing the upper boundary of the integrable region.
5. *Accuracy factor.* A definite integral is computed using an iterative algorithm that can be driven to any finite degree of accuracy. The HP 48 can offer accuracy up to its 12-digit limitation. The integration function determines the accuracy factor from the current display format. For example, a STD setting mandates the search to the 12-digit limitation of the machine, while 2 FIX sets an accuracy factor of 0.01 or 1% (i.e. the search stops when the HP 48 finds a value of the integral to within 1% uncertainty); and a 5 FIX setting indicates a 0.00001 accuracy factor (0.001%).*

To use these five inputs to compute a definite integral on the HP 48, simply adjust the display setting to the appropriate accuracy, then do any one of the following:

1. Enter the other four inputs in the correct order (lower limit, upper limit, integrand, integration variable) onto the first four levels of the stack and press [→][∫][←][→NUM].
2. Press ['] to begin an algebraic expression, press [→][∫] to enter the integral function, enter the four inputs in the correct order (lower limit, upper limit, integrand, integration variable) separated by commas, and enter the expression onto the stack. Press [EVAL][←][→NUM] to evaluate the integral.

*Note that the integration algorithm doubles the number of points sampled—and hence the amount of time—for each successive iteration, so it's important to keep the number of iterations to the minimum necessary: don't use STD unless you really must, but don't use just 2 FIX if you truly need accuracy to within .001%. After any integration, the HP 48 computes the uncertainty of the integration result it reports and stores it in the reserved variable, IERR. Thus, after performing an integration, look in the [VAR] menu for IERR and press it if you want to know the uncertainty of the result.

3. Press [←][EQUATION] to begin the Equation Writer, then type in each of the four inputs in its appropriate spot, pressing [▶] to jump forward between inputs. Enter the finished expression onto the first level of the stack and press [EVAL][←][→NUM] to evaluate the integral.
4. Press [→][SYMBOLIC][ENTER] to display the **INTEGRATE** input form. Enter each of the four inputs into its appropriate field—the integrand in **EXPR:**, the integration variable in **VAR:**, the lower limit in **LO:**, and the upper limit in **HI:**. Change the **RESULT:** field to Numeric, if necessary, and press **OK** to compute the definite integral returning the result to the stack.*

The following examples illustrate the use of each of these methods and give you some practice computing definite integrals in the process.

Example: Compute the following integral using the direct stack method (#1):

$$\int_0^2 \frac{x}{\sqrt{4x^2+8}}\,dx$$

1. Enter the limits, lower limit first: [0][ENTER][2][ENTER].
2. Enter the integrand and variable of integration: ['][α][←][X][÷][√x] [←][()][4][×][α][←][X][y^x][2][+][8][ENTER]['][α][←][X][ENTER].
3. Fix the display to 4 places (i.e. .01% accuracy) and then compute the definite integral: [4][←][MODES] **FMT** **FIX** [→][∫][←][→NUM].

Result: 0.5176

*Note that this fourth method explicitly reminds you about the fifth input by displaying a **NUMBER FORMAT:** field to allow you to change the number format—and thus the accuracy factor—before computing the integral. None of the other three methods offer such a reminder.

Example: Compute this integral using the normal algebraic method (#2):

$$\int_{-\frac{\pi}{15}}^{\frac{14\pi}{15}} \frac{\cos x}{1+2\sin x}\,dx$$

1. Enter the algebraic integral expression: ['] [→] [∫] [+/−] [←] [π] [÷] [1] [5] [←] [,] [1] [4] [×] [←] [π] [÷] [1] [5] [←] [,] [COS] [α] [←] [X] [▶] [÷] [←] [()] [1] [+] [2] [×] [SIN] [α] [←] [X] [▶] [▶] [←] [,] [α] [←] [X] [ENTER].
2. Because the integrand include trigonometric functions, make sure that you're in RAD mode, then compute the integral (to 4 decimal places): [←] [RAD] (if necessary) [EVAL] [←] [→NUM].

Result: `0.4426`

Example: Compute this integral using the EquationWriter method (#3):

$$\int_{1}^{4} \frac{xe^{8x^2}}{e^{8x^3}+1}\,dx$$

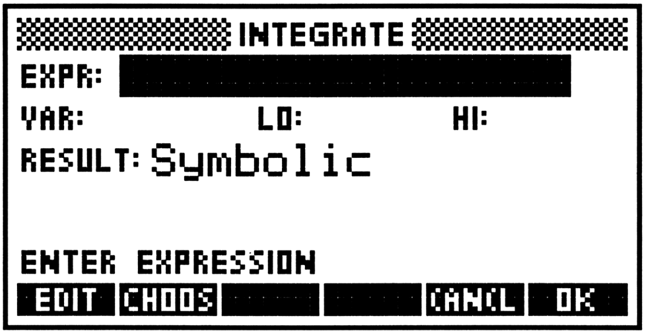
1. Enter the algebraic integral expression using the EquationWriter: [←] [EQUATION] [→] [∫] [1] [▶] [4] [▶] [α] [←] [X] [←] [e^x] [8] [α] [←] [X] [y^x] [2] [▶] [▶] [÷] [←] [e^x] [8] [α] [←] [X] [y^x] [3] [▶] [▶] [+] [1] [▶] [▶] [α] [←] [X] [ENTER].
2. Integrate (to 4 places): [EVAL] [←] [→NUM].

Result: `0.1004`

Example: Compute the following integral using the input form method (#4):

$$\int_2^3 \frac{\sqrt{8+x}}{\sqrt{8-x}}\,dx$$

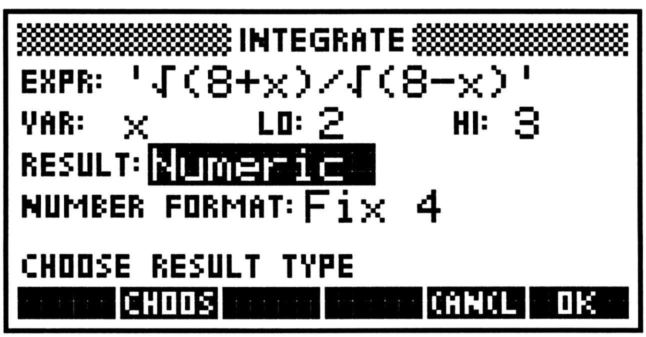
1. Open the INTEGRATE input form: [→] [SYMBOLIC] [ENTER]:



2. Enter the integrand in the EXPR: field: ['] [√x] [←] [( )] [8] [+] [α] [←] [X] [▶] [÷] [√x] [←] [( )] [8] [−] [α] [←] [X] [ENTER].

3. Enter the variable of integration and then the two limits: [α] [←] [X] [ENTER] [2] [ENTER] [3] [ENTER].

4. Change the RESULT: field to Numeric: [+/−]. Notice that an additional field then appears:



5. Change NUMBER FORMAT: to Fix 6, thereby changing the accuracy level of the computation: [▼] [▶] [6] [ENTER].

6. Compute the numeric definite integral: [OK].

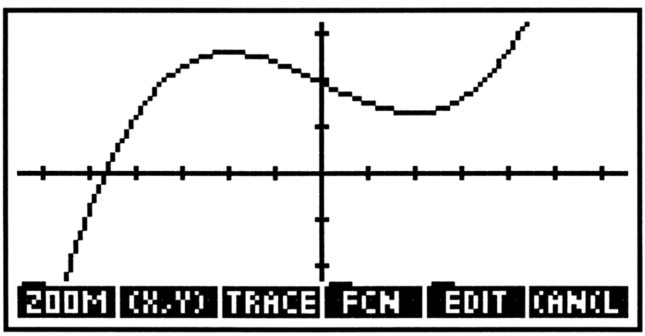
Result: 1.383500

Graphical Integration

Since integration computes the signed area* bounded by a curve, the x-axis, and two vertical lines, it's reasonable that the HP 48 offers a means of computing a definite integral *while viewing the plot of a curve.*

Example: Plot the curve $y = 2x^3 - 5x + 9$, and then compute the signed area between the curve and the x-axis and between $x = -3$ and $x = 2.4$:

1. Open the **PLOT** application, set **TYPE:** to Function and reset the plot parameters: [↱][PLOT][▲][α][F][DEL][▼][ENTER].
2. Highlight the **EQ:** field and enter the curve: [▼]['][2][X][α][↰][X][y^x][3][−][5][X][α][↰][X][+][9][ENTER].
3. Set **INDEP:** to x (lower-case), **H-VIEW** to -3 3, **V-VIEW** to -15 15, and leave the remaining plot parameters at their defaults. Then draw the plot: [α][↰][X][ENTER][3][+/−][ENTER][3][ENTER][▶][1][5][+/−][ENTER][1][5][ENTER] **ERASE** **DRAW**.



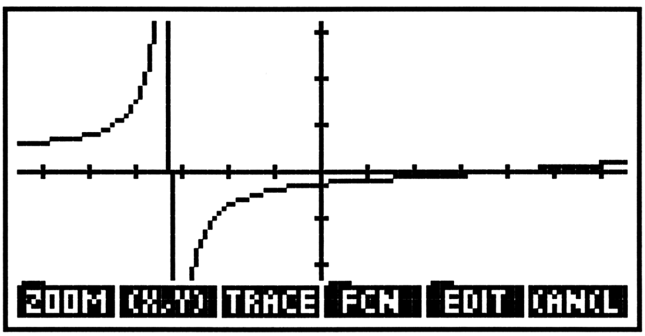
4. Press **TRACE** **(X,Y)** and move the cursor (using [◀]) to **X:-3.0000** at the left-hand side of the plot. Then acknowledge that point as the lower limit of an area computation: [NXT] **FCN** **AREA**.
5. Press [NXT] **PICT** **TRACE** **(X,Y)** and move the cursor (using [▶]) to the upper limit (at **X: 2.4000**). Then finish the area computation by pressing [NXT] **FCN** **AREA** again. Result: **AREA: 32.7888**

*Remember that "signed" area means that the area where the curve is "below" the x-axis is treated as a negative number and the area where the curve is "above" the x-axis is treated as a positive number.

One important point about all integration procedures: *Integration is defined only for intervals over which the given function is continuous.* If a discontinuity exists in the function between the limits given for a definite integral, the HP 48 will either generate an error message or return an incorrect result. Plotting a function before integrating is a good way to avoid this problem.

Example: Plot $f(x) = 4x^2 - \dfrac{5}{2x+1}$ to see if it is integrable over $-1 \le x \le 1$

1. Return to the **PLOT** input screen and enter the function in the **EQ:** field: [CANCEL] [▼] ['] [4] [×] [α] [←] [X] [y^x] [2] [−] [5] [÷] [←] [( )] [2] [×] [α] [←] [X] [+] [1] [ENTER].
2. Set **H-VIEW** to **-1 1** and **V-VIEW** to **-50 50** and plot the function: [▶] [1] [+/−] [ENTER] [1] [ENTER] [▶] [5] [0] [+/−] [ENTER] [5] [0] [ENTER] **ERASE** **DRAW**.



3. Notice the discontinuity at $x = -0.5$. The function is *not continuous* between -1 and 1 and is therefore *not* integrable over that interval. Try an integration anyway—to see what happens. Move the cursor to the left-hand edge of the plot (**X: -1.0000**) and press **FCN** **AREA**. Move the cursor to the right-hand edge (**X: 1.0000**) and press **AREA** again. Result: **AREA: (-0.0799,7.8540)**

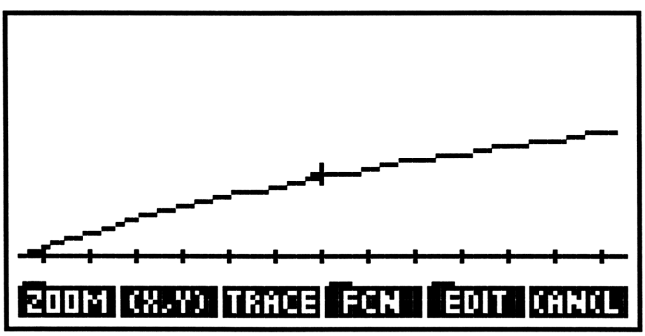
   In this case, because the integration algorithm never selected the exact discontinuity as a sample point, it didn't error, but instead gave a complex number value for the area—obviously incorrect. But be forewarned! It might just as easily have returned an incorrect *real* number.

The Area Function

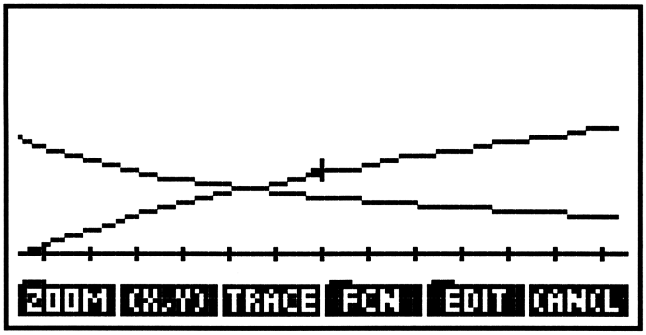
If you establish a fixed lower limit, say, $x = a$, and allow the upper limit to vary, the definite integral becomes a function where the independent variable is the upper limit and the dependent variable is the area of the defined region. This function is usually referred to as the *Area function*: $A(x) = \int_a^x f(t)dt$. Because it is a function, you can plot it, find its derivative, or use the Solve feature to solve the Area function for the value of the upper limit that generates a specific area.

Example: Plot the Area function, $A(x) = \int_1^x \frac{1}{t} dt$ for $1 \le x \le 3$, then its derivative.

1. Return to the PLOT screen and enter the Area function (a symbolic definite integral) in the EQ: field: [CANCEL] [▼] ['] [↱] [∫] [1] [↰] [,] [α] [↰] [X] [↰] [,] [1] [÷] [α] [↰] [T] [↰] [,] [α] [↰] [T] [ENTER].
2. Set H-VIEW to 1 3 and V-VIEW to -.5 2: [▶] [1] [ENTER] [3] [ENTER] [▶] [•] [5] [+/–] [ENTER] [2] [ENTER].
3. Plotting the Area function requires that an integral is computed for *every* sample point—and thus a very long time to plot the Area function. Minimize the delay by setting the number display format to 2 FIX and the step-size to 4 pixels (i.e. only one of every four pixels will be a sample point): [▼] [NXT] [CALC] [2] [α] [α] [F] [I] [X] [ENTER] [OK] [NXT] [OPTS] [▼] [▼] [4] [ENTER] [✓CHK].
4. Draw the plot: [OK] [ERASE] [DRAW].



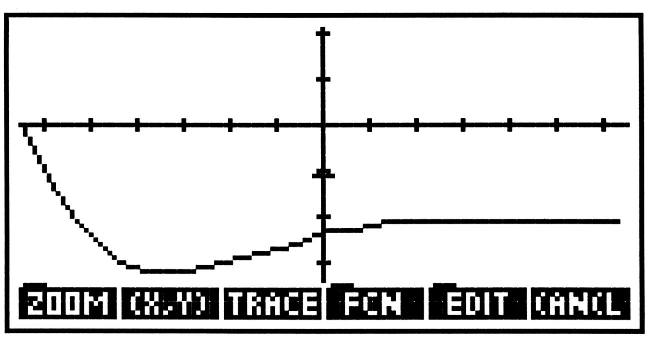
5. Now plot the derivative of the Area function: **FCN** [NXT] **F'**.



Hmm! The plot of the derivative looks familiar. Indeed, it's the same as the plot for $1/x$. The *derivative* of the area function is the *integrand* of the area function! This illustrates how you might plot an antiderivative of a given function. Simply plot the area function using the given function as the integrand!

Example: Plot the antiderivative of $\dfrac{\cos t}{2^t}$ over the interval -π to π.

1. Return to the **PLOT** screen, enter $\displaystyle\int_{-\pi}^{x} \frac{\cos t}{2^t}\,dt$ into **EQ:**, set **H-VIEW** to **-3.14 3.14** and **V-VIEW** to **-8 4**: [CANCEL] ['] [→] [∫] [+/–] [←] [π] [←] [,] [α] [←] [X] [←] [,] [COS] [α] [←] [T] [▶] [÷] [2] [y^x] [α] [←] [T] [←] [,] [α] [←] [T] [ENTER] [▶] [3] [.] [1] [4] [+/–] [ENTER] [3] [.] [1] [4] [ENTER] [▶] [8] [+/–] [ENTER] [4] [ENTER].
2. Since the number format and step-size adjustments are still in place from the previous example, draw the plot: **ERASE** **DRAW**.



Symbolic Integration: Pattern Matching

Finding the symbolic integral or antiderivative of a function on the HP 48 presents two kinds of problems. First, the integration command *always* requires that you specify lower and upper limits—which seems unnecessarily cumbersome if you simply want a formal *indefinite integral.* You can overcome this problem fairly easily with a simple program such as `INDEF` (see page 296) that uses dummy limits and then ignores the lower limit in the result (see page 173 for more details).

Secondly, there is no computer algorithm capable of finding the antiderivative of any general function comprised solely of the analytical functions set in the HP 48. Some can handle a reasonably large subset of such functions, but they require computing resources far beyond those of the HP 48. Thus, as a necessary compromise, the HP 48 limits its symbolic integration capabilities to a small range of all possible functions: polynomials, for which a simple antiderivative algorithm is available; and functions that match a list of patterns built into the HP 48's memory.

The table below lists all of the patterns that the HP 48 can match. Note that for each pattern listed, f is the variable of integration or *a linear function of the variable of integration.*

The HP 48 is quite picky about the form a function must have before it can be matched. Thus, given the pattern, $\frac{1}{\sin f \cos f}$, the HP 48 can find $\frac{1}{\sin x \cos x}$, $\frac{1}{\sin(2x)\cos(2x)}$, and $\frac{1}{\sin(2x+3)\cos(2x+3)}$, but not $\frac{1}{\cos x \sin x}$ (because the denominator terms are reversed), or $\frac{1}{\sin\left(x^2\right)\cos\left(x^2\right)}$ (because f isn't linear in x), or $\frac{1}{\sin(x)\cos(2x)}$ (because f is defined two distinct ways, x and $2x$), or $\frac{1}{2\sin x \cos x}$ (because of the coefficient in the denominator). Note, however, in this last case, that rewriting it as $\frac{1}{2}\left(\frac{1}{\sin x \cos x}\right)$ does allow the pattern to be matched after all. Here are the built-in patterns:

Pattern (Function)	*Replacement (Antiderivative)*
SQ(f)	f^3/3
√f	2*f^1.5/3
1/√f	2*√f
INV(√f)	2*√f
1/(2*√(f))	2*√(f)*.5
INV(2*√(f))	2*√(f)*.5
f^z (where z is symbolic)	IFTE(z==-1,LN(f),f^(z+1)/(z+1))
f^n (where n is real, ≠0,-1)	f^(n+1)/(n+1)
f^0	f
f^-1	LN(f)
1/f	LN(f)
INV(f)	LN(f)
COS(f)	SIN(f)
SIN(f)	-COS(f)
TAN(f)	-LN(COS(f))
ACOS(f)	f*ACOS(f)-√(1-f^2)
ASIN(f)	f*ASIN(f)+√(1-f^2)
ATAN(f)	f*ATAN(f)-LN(1+f^2)/2
COSH(f)	SINH(f)
SINH(f)	COSH(f)
TANH(f)	LN(COSH(f))
EXP(f)	EXP(f)
EXPM(f)	EXP(f)-f
LN(f)	f*LN(f)-f
LOG(f)	.434294481904*f*LN(f)-f
ALOG(f)	.434294481904*ALOG(f)
SIGN(f)	ABS(f)
TAN(f)^2	TAN(f)-f
1/TAN(f)	LN(SIN(f))
INV(TAN(f))	LN(SIN(f))
TAN(f)/COS(f)	INV(COS(f))
1/(SIN(f)^2)	-INV(TAN(f))
INV(SIN(f)^2)	-INV(TAN(f))
1/(COS(f)*SIN(f))	LN(TAN(f))
INV(COS(f)*SIN(f))	LN(TAN(f))
1/(SIN(f)*COS(f))	LN(TAN(f))
INV(SIN(f)*COS(f))	LN(TAN(f))

Pattern (Function)	*Replacement (Antiderivative)*
1/(SIN(f)*TAN(f))	-INV(SIN(f))
INV(SIN(f)*TAN(f))	-INV(SIN(f))
1/(TAN(f)*SIN(f))	-INV(SIN(f))
INV(TAN(f)*SIN(f))	-INV(SIN(f))
TANH(f)/COSH(f)	INV(COSH(f))
1/TANH(f)	LN(SINH(f))
INV(TANH(f))	LN(SINH(f))
1/(COSH(f)^2)	TANH(f)
INV(COSH(f)^2)	TANH(f)
1/(SINH(f)^2)	-INV(TANH(f))
INV(SINH(f)^2)	-INV(TANH(f))
1/(COSH(f)*SINH(f))	LN(TANH(f))
INV(COSH(f)*SINH(f))	LN(TANH(f))
1/(SINH(f)*COSH(f))	LN(TANH(f))
INV(SINH(f)*COSH(f))	LN(TANH(f))
1/(SINH(f)*TANH(f))	-INV(SINH(f))
INV(SINH(f)*TANH(f))	-INV(SINH(f))
1/(TANH(f)*SINH(f))	-INV(SINH(f))
INV(TANH(f)*SINH(f))	-INV(SINH(f))
1/(1-f^2)	ATANH(f)
INV(1-f^2)	ATANH(f)
1/(1+f^2)	ATAN(f)
INV(1+f^2)	ATAN(f)
1/(f^2+1)	ATAN(f)
INV(f^2+1)	ATAN(f)
1/(√(f-1)*√(f+1))	ACOSH(f)
INV(√(f-1)*√(f+1))	ACOSH(f)
1/√(1-f^2)	ASIN(f)
INV(√(1-f^2))	ASIN(f)
1/√(1+f^2)	ASINH(f)
INV(√(1+f^2))	ASINH(f)
1/√(f^2+1)	ASINH(f)
INV(√(f^2+1))	ASINH(f)

Try the following examples to see how symbolic pattern-matching works:

Example: Evaluate $\int_a^b x + \cos x \, dx$.

1. Enter the integral: CANCEL CANCEL ' ⮫ ∫ α ⬅ A ⬅ , α ⬅ B ⬅ , α ⬅ X + COS α ⬅ X ▶ ⬅ , α ⬅ X ENTER.
2. Purge the variables a and b to assure a symbolic result, then evaluate: ⬅ { } α ⬅ A SPC α ⬅ B ENTER ⬅ PURG EVAL EVAL.

   Result: `'SIN(b)+b^2/2-(SIN(a)+a^2/2)'`

Note that the integrand is divided into its additive terms before searching for a match. That is, the HP 48 treats the integral in this example as $\int_a^b x \, dx + \int_a^b \cos x \, dx$.

It computes the first integral using its polynomial rules and the second by matching a built-in pattern. Since *all* terms can be evaluated, the symbolic integration is successful. If any one of the terms in the integral can't be matched, then the HP 48 returns that part of the integral expression unevaluated.

The next example illustrates that the pattern-matching finds linear functions of the integration variable.

Example: Evaluate $\int_a^b x + \cos(2x - 5) \, dx$ and $\int_a^b x + \cos x^2 \, dx$.

1. The first integral: ' ⮫ ∫ α ⬅ A ⬅ , α ⬅ B ⬅ , α ⬅ X + COS 2 × α ⬅ X − 5 ▶ ⬅ , α ⬅ X ENTER EVAL EVAL.

   Result: `'SIN(2*b-5)/2+b^2/2-(SIN(2*a-5)/2+a^2/2)'`
2. The second integral: ' ⮫ ∫ α ⬅ A ⬅ , α ⬅ B ⬅ , α ⬅ X + COS α ⬅ X y^x 2 ▶ ⬅ , α ⬅ X ENTER EVAL EVAL.

   Result: `'b^2/2-a^2/2+∫(a,b,COS(x^2),x)'`

   Notice how the HP 48 evaluates as much of the integrand as it can, but leaves the part for which it finds no match as an unevaluated integral expression.

The HP 48 approach to symbolic integration is also flexible with respect to constants that can be removed from the integral expression. (Remember that integration treats all variables other than the integration variable as constants.)

Example: Evaluate the following integrals: $\int_a^b p \cos x \, dx$ and $\int_a^b x \cos x \, dx$

1. Enter the first integral: ['] [→] [∫] [α] [←] [A] [←] [,] [α] [←] [B] [←] [,] [α] [←] [P] [×] [COS] [α] [←] [X] [▶] [←] [,] [α] [←] [X] [ENTER].
2. Evaluate it: [EVAL] [EVAL].

   Result: `'p*SIN(b)-p*SIN(a)'`

   The HP 48 removes the constant p from the integrand before attempting to match the remainder—and is successful.
3. Enter the second integral: ['] [→] [∫] [α] [←] [A] [←] [,] [α] [←] [B] [←] [,] [α] [←] [X] [×] [COS] [α] [←] [X] [▶] [←] [,] [α] [←] [X] [ENTER].
4. Evaluate it: [EVAL] [EVAL].

   Result: `'∫(a,b,x*COS(x),x)'`

   This time, the factor x is treated (quite properly) not as a constant but as the variable of integration. The HP 48 cannot find a match for the integrand and therefore returns the unevaluated integral expression.

Enhancing the Built-In Integration Tools

As you've seen, the integration tools built into the HP 48 are quite useful but not adaptable to all of your needs. Fortunately, you can add many useful extensions and enhancements via programs. The remaining sections of this chapter demonstrate the programmed integration enhancements included with this book:

Numeric

- Approximating definite integrals using methods different than the built-in algorithm (Riemann sums, Simpson's Rule, etc.).
- Approximating the definite integral from a set of data for which the underlying function is unknown.
- Improving computational speed and/or accuracy by segmenting a definite integral.
- Accurately computing definite integrals with one or more infinite limits.
- Accurately computing definite integrals whose integrand cannot be evaluated at one of the limits.
- Accurately computing the definite integral over a range containing one or more discontinuities.

Graphical

- Plotting the specific antiderivative of a given function with known initial conditions.

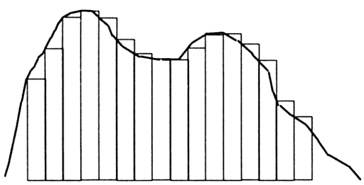
Symbolic

- Simplifying the computing of an indefinite integral.
- Adding to the patterns that can be matched using symbolic integration.
- Simplifying the integrand using segmentation.
- Simplifying the integrand using the method of substitution.
- Simplifying the integrand using integration by parts.
- Simplifying the integrand using partial fraction expansion.

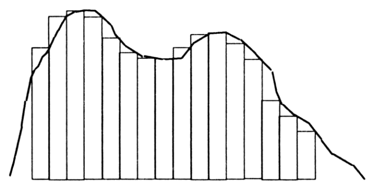
Approximating the Definite Integral of a Given Function

All methods of approximating a function's definite integral are based on computing the limiting value for an infinite sequence of approximations, each of which can be computed exactly. Specifically, the definite integral—the area under a given curve between two limits—is approximated by summing the areas of a sequence of rectangles whose height matches that of the curve at some point within the width of each rectangle.

Look at the three most common rectangular approximations:



Left Rectangular



Right Rectangular

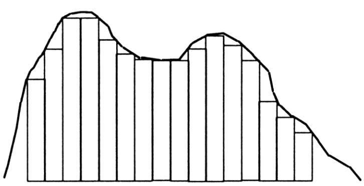
Mid Rectangular

The Left Rectangular approximation uses rectangles whose height matches the value of the function at the *left* edge of the each rectangle; the Right Rectangular approximation uses rectangles whose height matches the value of the function at the *right* edge of the each rectangle; the Mid Rectangular approximation uses rectangles whose height matches the value of the function at the value *midway* between the right and left edges of each rectangle.

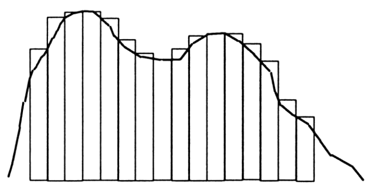
All of the approximations have irregular wedge-shaped "errors"—some of which overstate the area under the curve and some of which understate the area under the curve. But notice that using more rectangles—by decreasing their widths—reduces the errors; the approximations become more accurate. And the smaller the width, the closer these approximations are to each other—as well as to the actual area under the curve.

Thus, the definite integral is the limit of the sum of the areas of these rectangles as their widths approach zero. In practice, of course, you can't compute using an infinite number of rectangles of zero width, so approximating a definite integral means deciding how precise you want to get and then deciding what number of rectangles (and thus their widths) you need to achieve that precision.

The sum of a set of rectangular areas used to approximate a definite integral is known as a *Riemann sum*. Riemann sums differ from one another in two ways: the width of each rectangle (also known as *partition size*) and the rule used to determine the height of each rectangle. You've already seen three rules for Riemann sum rectangle heights: Left Rectangular, Right Rectangular, and Mid Rectangular. But there are two other important rules: use the smallest value of the function over each subinterval (*Lower Riemann*); and use the largest value of the function over each subinterval (*Upper Riemann*):



Lower Riemann



Upper Riemann

Notice that the Lower Riemann sum is less than the actual definite integral (since all of the "error wedges" represent underestimations of the function); and likewise, the Upper Riemann sum is greater than the actual definite integral (as those "error wedges" represent overestimations of the function).

Also, notice that for intervals over which a function always increases, the Lower Riemann sum is the same as the Left Rectangular sum (lowest value of function is at left edge) and the Upper Riemann sum is the same as the Right Rectangular sum (highest value of function is at right edge).

The Lower and Upper Riemann sums are primarily of use as theoretical lower and upper bounds of the definite integral. They are not used actually to compute an estimate because of the extra computational time it would take to find the maximum and minimum values of the function within each subinterval. Instead, the Left, Right, and Mid Rectangular sums are used because they are easier to compute. Usually (though not always) the Mid Rectangular sum is the best of the three approximations.

However, there are two other commonly used rules for Riemann sum approximations that are weighted averages of the three basic Riemann sums:

- *Trapezoidal* = 1/2 of Left + 1/2 of Right (i.e. the average of the Left and Right Rectangular sums).
- *Simpson's* = 1/6 of Left + 2/3 of Mid + 1/6 of Right.

Simpson's rule is derived from the observation that, while the Mid Rectangular sum and Trapezoidal each have their strengths and weaknesses as estimates, on average the actual definite integral lies between the two, but twice as close to the Mid Rectangular estimate than to the Trapezoidal estimate. Simpson's rule is thus the most refined approximation of the five mentioned thus far for a given number of rectangles.

The five Riemann sum estimates mentioned above can be computed by hand if the number of rectangles used isn't very large. But of course, it's a useful task for a program as well. DEFINT (see page 292) takes an algebraic expression of a definite integral from level 2 and the number of rectangles from level 1 and returns a list of labeled estimates for the definite integral to level 1. The returned list contains the five Riemann sums followed by the estimate using the built-in integration routine (which is a refinement of Simpson's rule, known as Romberg's method—more efficient for machine computation). The estimates are rounded to the number of digits that reflects their precision given the number of rectangles you used (keep in mind, as always, that larger numbers of rectangles require increasing time to compute).

Example: Use 40 rectangles to find the Left, Right, and Mid Rectangular estimates of $\int_{-\frac{1}{2}}^{1} \tan^{-1} x \, dx$.

1. Enter the algebraic integral expression: ['] [→] [∫] [+/−] [•] [5] [←] [,] [1] [←] [,] [←] [ATAN] [α] [←] [X] [▶] [←] [,] [α] [←] [X] [ENTER].
2. Enter the number of approximating rectangles: [4] [0] [ENTER].
3. Execute the DEFINT program to compute the approximations: [α] [α] [D] [E] [F] [I] [N] [T] [ENTER] or [VAR] ([NXT] or [←] [PREV]) [DEFIN].

   Result: { :left: .29512 :mid: .31859 :right: .34196 :trap: .31854 :simp: .31857 :intg: .31857 }

   The Left Rectangular estimate is 0.29512; the Right Rectangular estimate is 0.34196; the Mid Rectangular estimate is 0.31859; etc.

Example: Use 25 rectangles to calculate the Trapezoidal and Simpson's estimates of $\int_{\frac{1}{2}}^{\frac{9}{2}} \sin x^3 \, dx$.

1. Enter the algebraic integral expression: ['] [→] [∫] [•] [5] [←] [,] [4] [•] [5] [←] [,] [SIN] [α] [←] [X] [y^x] [3] [▶] [←] [,] [α] [←] [X] [ENTER].
2. Enter the number of approximating rectangles: [2] [5] [ENTER].
3. Execute DEFINT to compute the approximations: [DEFIN].

   Result: { :left: .909 :mid: -.1274 :right: .886 :trap: .8975 :simp: .2143 :intg: .4473 }

   Notice that the estimates are all over the place. Periodic functions, such as this one, usually require many more rectangles that non-periodic functions to achieve the same level of accuracy.

4. Repeat the computation using 100 rectangles: [→] [UNDO] [◀] [1] [0] [0] [ENTER] [DEFIN].

   Result: { :left: .44108 :mid: .45226 :right: .43534 :trap: .43821 :simp: .44758 :intg: .44734 }

Approximating the Definite Integral of a Data Set

In many real-world situations, you may be faced with a set of data whose underlying *function* is unknown, but whose definite integral you need to estimate. How can you estimate a definite integral ("the area under the curve") if you *don't know the function you're integrating*, just a set of data that (presumably) represents it?

This requires that you make some assumptions about the underlying function—even though you may know little about it. There are two approaches to this—interpolation and regression. *Interpolation* methods create functions that actually contain every data point. *Regression* methods create functions that minimize the accumulated distances between themselves and their data points.

Regression assumes that there is some measurement error in the data set and that you want to find an underlying function that best approximates the fundamental relationship of the variables while largely ignoring fluctuations due to measurement error. Interpolation, on the other hand, assumes that there's no measurement error in the data set, that the points are exact and that all variation is due to the underlying function and none of it due to error.

This section describes four particular methods of approximating a definite integral from a set of data, each of which has its own program:

1. ΣINT1 bases its estimate on an interpolation called *linear piecewise*, constructed by connecting each data point to its nearest neighbor with a line segment (i.e. drawing "dot-to-dot"). The area under this "curve" is then divided into a series of trapezoids whose areas are computed and summed.

2. ΣINT2 constructs a single polynomial that exactly contains all of the data points (i.e. an *interpolation*) and then integrates this function.

3. ΣINT3 uses a cubic spline interpolation, whereby each pair of neighboring data points are connected by a smooth portion of a third-degree polynomial (whose function may be different for each pair), to create a function whose definite integral is then estimated using Simpson's rule.

4. ΣINT4 allows you to input a *model* for a regression curve that you believe represents the underlying function that your data is manifesting. The program then computes the specific coefficients that minimizes the sum of squares of the distances between each data point and the specified regression curve (i.e. a *least-squares estimate*). The regression curve and the data (shown including their measurement errors) are plotted so that you can view the "goodness-of-fit." If the fit isn't very good, you may try another iteration of the same model or even try a new model. When you are satisfied with your regression curve, it is used to estimate the definite integral.

The methods of each of these programs have their strengths and weaknesses. None can be guaranteed to give you good estimates in all cases. Look at each in turn as you work through the following examples.

ΣINT1: Piecewise Linear Interpolation

ΣINT1 (see page 296) is by far the speediest of the programs, and its accuracy depends more on the number of data points you have than on the shape of the underlying curve. If you have a large data set, this will usually give you a good estimate in a relatively short time. Other methods may improve accuracy but take much longer to do so.

ΣINT1 assumes that the lower and upper limits of the definite integral coincide with two of your data points. Level 3 should contain a data matrix containing at least two columns (one for the independent variable and one for the dependent variable). The program treats the smallest value in the independent variable column as the lower limit for the integral and the largest value in the column as the upper limit for the integral. Level 2 should contain the column number for the independent variable and level 1 should contain the column number for the dependent variable. The definite integral estimate is returned to level 1.

Try an example of ΣINT1....

Example: Use a piecewise linear interpolation to estimate the definite integral for the underlying function (between 0 and 6) represented by the following data:

x	$f(x)$	x	$f(x)$	x	$f(x)$
0.0	19.44	2.1	5.45	4.2	22.86
0.3	23.92	2.4	-0.32	4.5	22.91
0.6	20.09	2.7	3.93	4.8	22.02
0.9	18.51	3.0	5.76	5.1	21.86
1.2	16.73	3.3	7.02	5.4	13.35
1.5	9.12	3.6	13.81	5.7	12.22
1.8	8.78	3.9	19.33	6.0	8.07

1. Enter the data as a two-column matrix with the x-values in column 1 and the $f(x)$-values in column 2 and store a copy as DS1:

   [↱][MATRIX]
   [0][SPC][1][9][•][4][4][ENTER][▼][•][3][SPC][2][3][•][9][2][ENTER]
   [•][6][SPC][2][0][•][0][9][ENTER][•][9][SPC][1][8][•][5][1][ENTER]
   [1][•][2][SPC][1][6][•][7][3][ENTER][1][•][5][SPC][9][•][1][2][ENTER]
   [1][•][8][SPC][8][•][7][8][ENTER][2][•][1][SPC][5][•][4][5][ENTER]
   [2][•][4][SPC][•][3][2][+/−][ENTER][2][•][7][SPC][3][•][9][3][ENTER]
   [3][SPC][5][•][7][6][ENTER][3][•][3][SPC][7][•][0][2][ENTER]
   [3][•][6][SPC][1][3][•][8][1][ENTER][3][•][9][SPC][1][9][•][3][3][ENTER]
   [4][•][2][SPC][2][2][•][8][6][ENTER][4][•][5][SPC][2][2][•][9][1][ENTER]
   [4][•][8][SPC][2][2][•][0][2][ENTER][5][•][1][SPC][2][1][•][8][6][ENTER]
   [5][•][4][SPC][1][3][•][3][5][ENTER][5][•][7][SPC][1][2][•][2][2][ENTER]
   [6][SPC][8][•][0][7][ENTER][ENTER][ENTER]
   ['][α][α][D][S][1][ENTER][STO]

2. Enter the column number for the independent variable: [1][ENTER].
3. Enter the column number for the dependent variable: [2][ENTER].
4. Estimate the definite integral for this data set using piecewise linear interpolation (ΣINT1): [VAR] (then [NXT] or [↰][PREV] as needed) **ΣINT1**. <u>Result</u>: 84.3315

ΣINT2: Single Polynomial Interpolation

When the underlying relationship may be polynomial in nature and you have only a small number of data points (i.e. less than 10), single polynomial interpolation will provide a reasonable estimate. But if the relationship isn't polynomial, this approach can give wildly inaccurate estimates ("let's be careful out there")!

ΣINT2 (see page 297) takes a data matrix from level 4 that contains at least two columns (one for the independent variable and one for the dependent variable), the column number of the independent variable from level 3, the column number of the de-pendent variable from level 2, and a list containing, in order, the lower and upper limits for the definite integral. The program returns the estimate of the definite integral to level 1. ΣINT2 uses Joseph Horn's PFIT routine (see page 311), included here with his permission, to compute the interpolating polynomial.

Example: Use single-polynomial interpolation to estimate the definite integral (between 5 and 50) of the underlying function represented by:

x	$f(x)$	x	$f(x)$	x	$f(x)$
5	34.8	10	134.7	14	159.3
18	156.9	24	132.2	30	117.4
35	132.5	41	186.6	50	342.2

1. Enter the data as a 2-column matrix (the x-values in column 1 and $f(x)$-values in column 2), and store a copy as DS2: [→][MATRIX][5] [SPC][3][4][•][8][ENTER][▼][1][0][SPC][1][3][4][•][7][ENTER][1][4][SPC] [1][5][9][•][3] [ENTER] [1][8] [SPC] [1][5][6][•][9] [ENTER] [2][4] [SPC] [1][3][2][•][2] [ENTER] [3][0] [SPC] [1][1][7][•][4] [ENTER] [3][5] [SPC] [1][3][2][•][5] [ENTER] [4][1] [SPC] [1][8][6][•][6] [ENTER] [5][0] [SPC] [3][4][2][•][2][ENTER][ENTER][ENTER]['][α][α][D][S][2][ENTER][STO].
2. Enter the column numbers for the independent and dependent variables: [1][ENTER] [2][ENTER].
3. Enter the limits of integration: [←][{ }][5][SPC][5][0][ENTER].
4. Estimate the definite integral using single polynomial interpolation (ΣINT2): [VAR] (then [NXT] or [←][PREV] as needed) **ΣINT2**.

Result: 7144.83213303

ΣINT3: Cubic Spline Interpolation

Cubic spline interpolation is a piecewise interpolation like that of ΣINT1, with two differences: Adjacent points are connected by segments of third-degree polynomials instead of line segments; and the curve created by cubic spline interpolation is continuous at each data point (because the "ending" slope of each polynomial segment matches the "starting" slope of the next one), while linear interpolation is discontinuous. Cubic spline interpolation performs well for most data sets, generally better than linear piecewise for smaller sets and worse for larger—a good choice for a small set whose underlying shape is unknown or questionable.

The program ΣINT3 (page 297) uses SPLINE (page 326) and SPLEVAL (page 326) to create and evaluate, respectively, the cubic spline interpolation. It computes the definite integral by iterating Simpson's rule estimations with increasing numbers of points until the result is obtained with at least four significant digits. ΣINT3 takes a data matrix from level 3 containing at least two columns (independent and dependent variables), the column numbers of the independent and dependent variables from levels 2 and 1, respectively. The smallest and largest values in the independent variable column become the lower and upper limits of integration, respectively. The estimate is returned to level 1.

Example: Using cubic spline interpolation, estimate the definite integral (between -5 and 5) of the underlying function for the following data:

x	f(x)	x	f(x)	x	f(x)
-5	12.55	-4	7.24	-3	2.42
-2	0.14	-1	-1.64	0	-2.88
1	-1.75	2	-0.28	3	2.59
4	7.35	5	12.55		

1. Enter the data and store as DS3: [→][MATRIX][5][+/–][SPC]... (etc.)... [1][2][•][5][5][ENTER][ENTER][ENTER]['][α][α][D][S][3][ENTER][STO].
2. Enter the column number for the independent variable: [1][ENTER].
3. Enter the column number for the dependent variable: [2][ENTER].
4. Estimate the integral via cubic spline interpolation (ΣINT3): [VAR] (then [NXT] or [←][PREV] as needed) [ΣINT3]. Result: 25.225705504

ΣINT4: Least-Squares Fit

A *fitted* curve does not "connect" the data points; rather, it represents a curve (whose general shape you determine) that approximates the data points—thereby assuming that the actual data points contain some amount of measurement error that accounts for the distance of the actual data from the fitted curve. The "goodness-of-fit" of a particular curve can be measured by totaling the squares of the "error" distances between the actual data points and the fitted curve. Fitting (as opposed to interpolating) a curve to a set of data is probably the best choice whenever you have strong analytic evidence about what the underlying model should be for a set of data, or when you have a lot of data points but are confident that the underlying function has a much smaller number of terms.

The program DATFIT (see page 291), which is used by ΣINT4, computes the fitted curve whose goodness-of-fit sum is smallest—the *least-squares fit*—for the particular general model being used. To fit a curve to a particular data set using DATFIT, you will need: the data set, the general model of curve you wish to fit, and an estimate of the measurement errors for each of the data points. DATFIT takes the data matrix containing at least two columns (one for the independent variable and one for the dependent variable) from level 4, the column number of the independent variable from level 3, the column number of the dependent variable from level 2, and an estimate of the measurement errors from level 1. The level 1 entry may either be a single real number reflecting equal measurement errors for all data points or a vector of measurement errors arranged in the same order as the data, one per point.

DATFIT will then display an input screen, prompting for 3 items: **(i)** a general model using only variable names for the coefficients and 'x' as the independent variable; **(ii)** a list of the coefficients in the model whose least-squares fitted values you seek; and **(iii)** an initial estimate of the coefficient values to begin the search procedure. Note that, for some non-linear models, good beginning esti-mates are necessary to assure that the best fit is found.

Finally, after you supply the requested inputs, DATFIT computes a fitted curve using your model and plots the curve along with the original data points—shown with their error bars (vertical lines whose length represents the size of the errors for the points) to allow you to visually inspect the goodness-of-fit for your model.

After pressing CANCEL to leave the plot, you have three options:

1. Press CANCL to accept the fitted curve. You will return to the stack to find the covariance matrix for the fit on level 3, the fitted curve expression on level 2, and a list of labeled coefficient values with their estimated standard deviations on level 1.
2. Press OK to refine the fit. The current estimates for the fitted coefficients are used as initial values and the fit is recomputed, then replotted against the data points. You may thus refine a particular fit as many times as you wish, although you will soon find that additional refinements make increasingly tiny differences and that such tiny differences aren't worth the time needed for the computations.
3. Enter a different model, list of coefficient names, and starting estimates, then press OK. This option is available to you in case your first estimate model was not very good and you wish to start over.

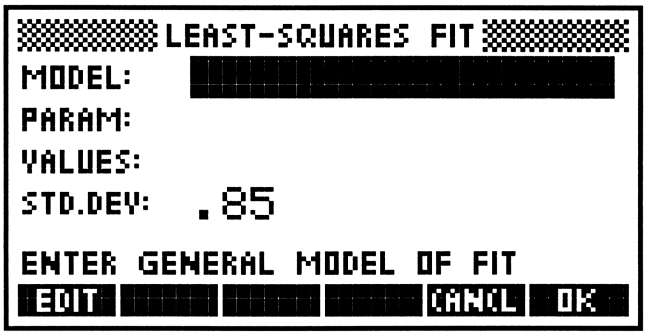
These three options are available each time you leave the plot of the fitted curve and the data points with their error bars. When you're satisfied with the fit, use option 1 to exit DATFIT.

The program ΣINT4 (see page 298) requires the same four inputs as DATFIT, plus an additional input on level 1. The data matrix is taken from level 5, the column number for the independent variable from level 4, the column number for the dependent variable from level 3, the measurement error estimate from level 2, and a list of the lower and upper limits of integration from level 1. ΣINT4 then executes DATFIT and, after exiting DATFIT, returns the estimate for the definite integral to level 1.

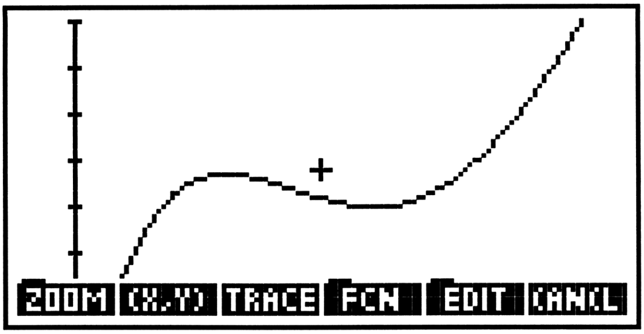
The three following examples illustrate the use of DATFIT and ΣINT4 using the three data sets you've already used for the three previous examples. The examples below assume that you have stored each in its appropriate name—DS1, DS2, or DS3—as you worked the previous examples. If you don't have these data sets stored, you'll need to enter them as new matrices instead.

Example: Using DS2, compute a least-squares fitted curve using a fourth-degree polynomial as the fit model. Assume the measurement error for all data points is 0.85. Then use the fitted curve to estimate the definite integral for the data between 5 and 50.

1. Recall the DS2 data matrix to level 1: [α][α][D][S][2][ENTER].
2. Enter the column numbers for the independent and dependent variables, respectively: [1][ENTER][2][ENTER].
3. Enter the estimated measurement error (a single real number, in this case): [•][8][5][ENTER].
4. Enter the limits of integration: [←][{ }][5][SPC][5][0][ENTER].
5. Begin ΣINT4: [VAR] ([NXT] or [←][PREV]) ΣINT4. Soon you'll see:



6. In the MODEL: field, enter the general form of the model you're using to fit the data—a 4th-degree polynomial ($Ax^4 + Bx^3 + Cx^2 + Dx + E$), in this case: ['][α][A][×][α][←][X][y^x][4][+][α][B][×][α][←][X][y^x][3][+][α][C][×][α][←][X][y^x][2][+][α][D][×][α][←][X][+][α][E][ENTER].
7. In the PARAM: field, enter the list of parameter names used in the general model: [←][{ }][α][A][SPC][α][B][SPC][α][C][SPC][α][D][SPC][α][E][ENTER].
8. In the VALUES: field, enter a list of initial guesses as to the values of the corresponding parameters. Use zeroes here (although for non-linear models its often necessary to use thoughtful initial estimates for parameters): [←][{ }][0][SPC][0][SPC][0][SPC][0][SPC][0][ENTER].
9. Compute the first-iteration values for the parameters and plot the resulting model against the data points: OK....



The fitted curve looks to match the data so well that it overlays nearly all of the data bars!

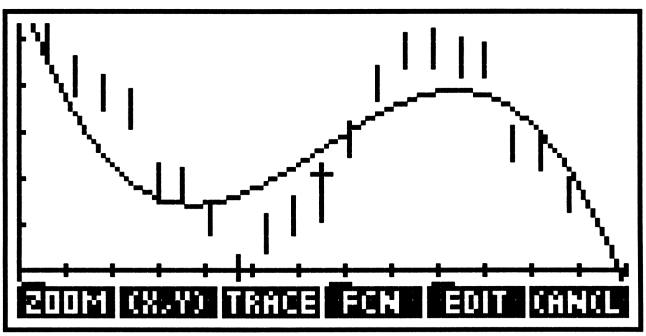
10. Press CANCEL to return to the **LEAST-SQUARES FIT** screen. You'll see that the initial parameter values you entered have been replaced with the computed values, and the standard deviation for the fitted model has been recomputed.

11. Although you could compute and plot another iteration, if necessary, it's obviously not necessary here judging by the plot and the fact that the computed standard deviation is less than the original measurement error. Instead, accept the current model and use it to compute the integral: CANCEL. Result: 7189.01807954

This value agrees within 1% with the value determined by single polynomial interpolation on page 145—probably because a polynomial model seems to be an appropriate one for this data set.

Example: Using DS1, compute a least-squares fitted curve using a model of your choosing, modifying it if necessary. Assume the measurement error for all data points is 1.75. After finding a good fitting curve, use it to estimate the definite integral for the data between 0 and 6.

1. Recall the DS1 data matrix to level 1: [α] [α] [D] [S] [1] [ENTER].
2. Enter the column numbers for the independent and dependent variables, respectively: [1] [ENTER] [2] [ENTER].
3. Enter the estimated measurement error: [1] [•] [7] [5] [ENTER].
4. Enter the limits of integration: [←] [{ }] [0] [SPC] [6] [ENTER].
5. Begin ΣINT4: [VAR] (then [NXT] or [←] [PREV] as needed) [ΣINT4].
6. In MODEL:, enter a model. The data suggests that a third-degree polynomial ($\text{A}x^3 + \text{B}x^2 + \text{C}x + \text{D}$) or a sine curve ($\text{A}\sin(\text{B}x + \text{C}) + \text{D}$) may be useful. Try the polynomial: ['] [α] [A] [×] [α] [←] [X] [y^x] [3] [+] [α] [B] [×] [α] [←] [X] [y^x] [2] [+] [α] [C] [×] [α] [←] [X] [+] [α] [D] [ENTER].
7. In the PARAM: field, enter the list of parameter names used in the general model: [←] [{ }] [α] [A] [SPC] [α] [B] [SPC] [α] [C] [SPC] [α] [D] [ENTER].
8. In the VALUES: field, enter a list of zeroes as initial guesses: [←] [{ }] [0] [SPC] [0] [SPC] [0] [SPC] [0] [ENTER].
9. Compute and plot the first-iteration: [OK]



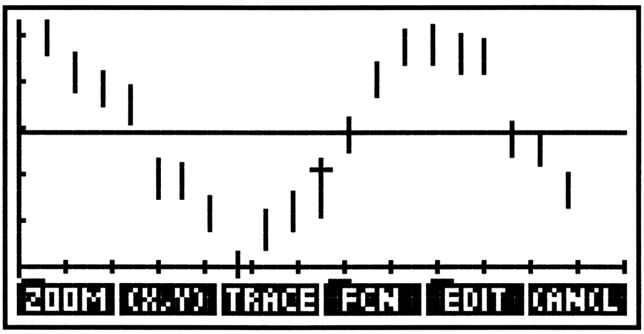
This time the fit doesn't seem very good: the curve touches just one-third of the error-bars and, at least on the left-hand side, seems to be concave where the data pattern appears convex.

10. Press [CANCEL] to return to the **LEAST-SQUARES FIT** screen. Note that the computed standard deviation is three times the measurement error—further evidence that you need a better fit.

At this point, you have three choices: **(i)** iterate the model a second time using the first iteration values as a starting point; **(ii)** input different starting values and recompute a first iteration using the same model; or **(iii)** start over with a different model. Experience will be your best teacher in these cases. This time, choice 3 beckons.

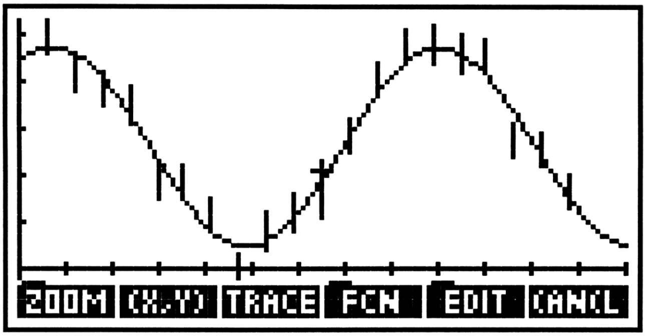
Highlight the **MODEL:** field and input the general sine curve: ['] [α] [A] [×] [SIN] [α] [B] [×] [α] [←] [X] [+] [α] [C] [▶] [+] [α] [D] [ENTER].

11. Enter zeroes for the values in the **VALUES:** field and compute the first iteration: [▼] [←] [{ }] [0] [SPC] [0] [SPC] [0] [SPC] [0] [ENTER] **OK**.



Hmmm... a straight line! Perhaps using zeroes as starting values wasn't a very good idea. Press **(X,Y)** and move the cursor around to get an estimated feel for the amplitude (A), period (2π/B), phase shift (-C/B), and vertical shift (D). Normally, the sine curve oscillates around the *x*-axis ($y = 0$). The sine curve formed by the data error-bars seem to oscillate around $y = 12.5$ so $D \approx 12.5$. The amplitude (half the distance between trough and peak) ≈ 10. The period ≈ 4.1 so $B \approx 6.3/4.1 \approx 1.5$. The phase shift ≈ -.6, so $C \approx 0.9$.

12. Return to **LEAST-SQUARES FIT** and enter the new set of initial guesses in **VALUES:** and recompute the first iteration: [CANCEL] [▼] [▼] [←] [{ }] [1] [0] [SPC] [1] [•] [5] [SPC] [•] [9] [SPC] [1] [3] [ENTER] **OK**....



Now you're in business! The fit looks to be pretty good, although there are a couple of data bars that don't touch the curve.

13. Press CANCEL to return to the set-up screen and notice that the standard deviation for the curve is nearly the same as the original measurement error, further indicating that the model and the fit are good. You can safely try another iteration using the current values as starting values, hoping to improve the fit a bit more. Press OK.

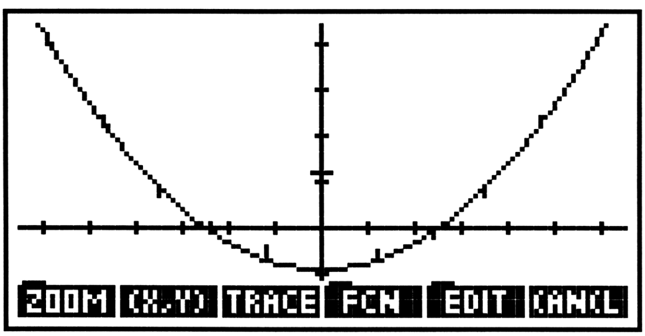
14. The resulting graph doesn't look much different than the previous iteration. Press CANCEL to see that the second one has improved the standard deviation only slightly. Unless you need very high precision values for the coefficients of the fitting model to achieve high precision in the integral, you probably have an adequate model with which to compute the integral and need no further iterations. Accept the model and compute the integral by pressing CANCEL.

<u>Result</u>: 84.3569752539

This differs from the speedy piecewise linear interpolation by less than 1/20 of a percent!

Example: Using DS3, compute a least-squares fitted curve using a quadratic polynomial. Assume the measurement errors for the data points are as follows: [0.2 0.3 0.3 0.4 0.4 0.4 0.3 0.3 0.2 0.2 0.1] (as the independent variable moves from -5 to 5). Use the best-fitting curve to estimate the definite integral for the data between -5 and 5.

1. Recall the DS3 data matrix to level 1: (α)(α)(D)(S)(3)(ENTER).
2. Enter the column numbers for the independent and dependent variables, respectively: (1)(ENTER)(2)(ENTER).
3. Enter the estimated measurement errors (as a vector of individual errors): (←)([ ])(•)(2)(SPC)(•)(3)(SPC)(•)(3)(SPC)(•)(4)(SPC)(•)(4)(SPC)(•)(4)(SPC)(•)(3)(SPC)(•)(3)(SPC)(•)(2)(SPC)(•)(2)(SPC)(•)(1) (ENTER).
4. Enter the limits of integration: (←)({ })(5)(+/−)(SPC)(5)(ENTER).
5. Begin ΣINT4: (VAR) (then (NXT) or (←)(PREV) as needed) ΣINT4.
6. In MODEL:, enter the form of the second-degree polynomial (Ax^2 + Bx+C): (')(α)(A)(×)(α)(←)(X)(y^x)(2)(+)(α)(B)(×)(α)(←)(X)(+)(α)(C)(ENTER).
7. In the PARAM: field, enter the list of parameter names used in the general model: (←)({ })(α)(A)(SPC)(α)(B)(SPC)(α)(C)(ENTER).
8. In VALUES:, enter zeroes: (←)({ })(0)(SPC)(0)(SPC)(0)(ENTER).
9. Compute the first-iteration values for the parameters and plot the resulting model against the data points: OK.



10. The fit looks very good. Press (CANCEL) to see that the standard deviation (.30095) is within the range of the initial measurement errors (.1 to .4). Accept the current model and use it to integrate: (CANCEL).
<u>Result</u>: 24.3993625484 (3% different than cubic spline, p. 146.)

Plotting and Solving with Definite Integrals

Although the HP 48 can compute numeric definite integrals in fairly reasonable amounts of time (assuming that the integrand is continuous over the integration interval), processes such as plotting and solving that require repeated computations of the definite integral are very, very slow, even if you set the numerical precision low (e.g. `2 FIX` or `3 FIX`).

For example, plotting the equation, $\int_{-\frac{\pi}{2}}^{x} \frac{t}{\pi} \cos t \, dt = x^3 - 4x - 2$, at default plot parameters and `3 FIX` requires nearly five minutes because the definite integral must be computed 131 times (once for each horizontal pixel). Similarly, finding the greatest positive solution to the equation (by moving the cursor near the rightmost intersection and using **FCN** **ISECT**) takes another half minute. `ISECT`, after all, uses the built-in root finder that must compute the definite integral on each iteration. If you need more precision than three places, then the time to compute the result increases dramatically.

A speedier alternative to using a definite integral directly in a plotting or solving context is to compute a Taylor's polynomial approximation (see page 42) of the integrand, evaluate the integral symbolically using the approximation instead of the original integrand, and substitute the result for the integral expression in the original equation. For this approach to work, the Taylor's approximation must converge with the original integrand in the region of the solution.

Example: Plot the equation $\int_{-\frac{\pi}{2}}^{x} \frac{t}{\pi} \cos t \, dt = x^3 - 4x - 2$, using a 9th-order Taylor's polynomial to approximate the integrand and evaluating the modified integral before plotting. Then find the greatest positive solution to the equation.

1. Enter the integrand (using a numeric value for π) and its independent variable: ['] [α][←][T] [ENTER][←][π] [←][→NUM] [÷] ['] [COS] [α][←][T] [ENTER][×]['][α][←][T][ENTER].
2. Enter the order and compute the Taylor's approximation and make an extra copy of the result: [9][ENTER][←][SYMBOLIC] **TAYLR** [ENTER]....

Result (to 2 places): '0.32*t-0.95/3!*t^3+1.59/5!*t^5
-2.23/7!*t^7+2.86/9!*t^9'.

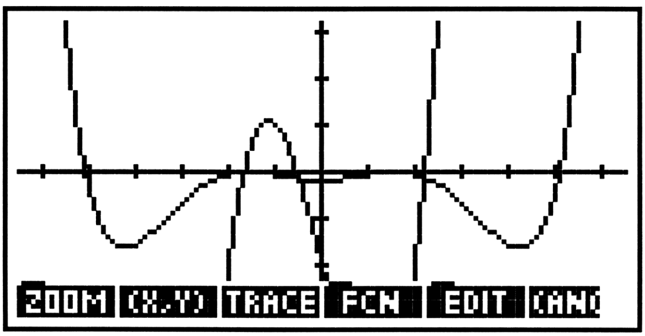
3. Now compute the integral using the Taylor's polynomial as the integrand: [←][π][←][→NUM][2][÷][SWAP]['][α][←][X][ENTER][SWAP]['][α][←][T][ENTER][→][∫][EVAL].

   Result (to 2 places): '7.89E-6*(x^10/10)-4.42E-4*
   (x^8/8)+0.01*(x^6/6)-0.16*
   (x^4/4)+0.32*(x^2/2)-0.18'

4. Enter the right-side of the original equation, set it equal to the previous result, and store the whole equation in 'EQ': ['][α][←][X][y^x][3][−][4][×][α][←][X][−][2][ENTER][←][=][←][PLOT][←][EQ].

5. Open the **PLOT** application, set **TYPE:** to Function and reset the plot parameters: [→][PLOT][▲][α][F][DEL][▼][ENTER].

6. The equation should already be showing in the **EQ:** field, so simply adjust the **INDEP:** variable and draw the plot: [▼][▼][α][←][X][ENTER][ERASE][DRAW].

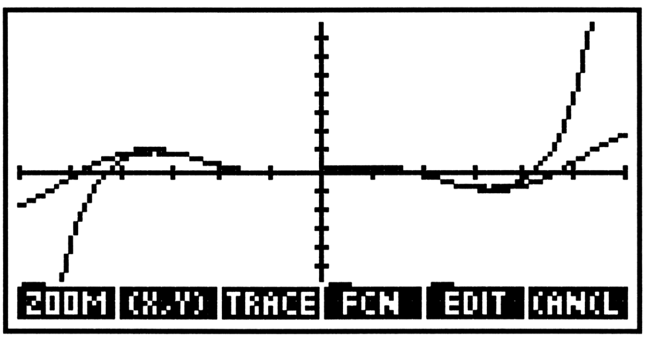


7. The two polynomials intersect each other three times. Move the cursor near and to the right of the positive most intersection point and press [FCN][ISECT]. Result (to 5 places): **X: 2.20283**

The accuracy of the result obtained using a Taylor's polynomial depends on how well the polynomial approximates the integrand near the solution. As you remember from Chapter 1 (page 45), you can view the interval of validity for a Taylor's approximation using the program PTAYL.

Example: Use PTAYL to determine whether the solution you found in the previous example falls within the interval of validity for the Taylor's approximation you used.

1. Return to the stack, and drop the result of the previous ISECT command so that the copy of the Taylor's approximation is on level 1 of the stack: [CANCEL] [CANCEL] [←].
2. Enter the original integrand and independent variable: ['] [α] [←] [T] [ENTER] [←] [π] [÷] ['] [COS] [α] [←] [T] [ENTER] [×] ['] [α] [←] [T] [ENTER].
3. Enter 0 (the point around which the approximation was centered) and 2.20283 (the point being approximated) and launch PTAYL: [0] [ENTER] [2] [.] [2] [0] [2] [8] [3] [ENTER] [VAR] ([NXT] or [←] [PREV] as needed) [PTAYL].



Thus, the result should be quite accurate. Indeed, although it takes 10 minutes of plotting and/or solving to determine, the most positive solution to the equation when solved directly using the integral expression is $x = 2.20282$—very good agreement indeed with the Taylor's polynomial approximation approach.

Numerically Estimating Difficult Integrals

Although the HP 48 can numerically integrate any definite integral—that is, it will eventually arrive at an answer—there are certain kinds of definite integrals that it finds difficult to compute either quickly or accurately (or both):

- An integral whose integrand contains a *cusp* (or "elbow point") within the interval of integration.
- An integral of finite value whose interval of integration is nevertheless unbounded (i.e. one or both limits are infinite).
- An integral with finite limits whose integrand is undefined at one of those limits.
- An integral with finite limits whose integrand contains a discontinuity within the integration interval.

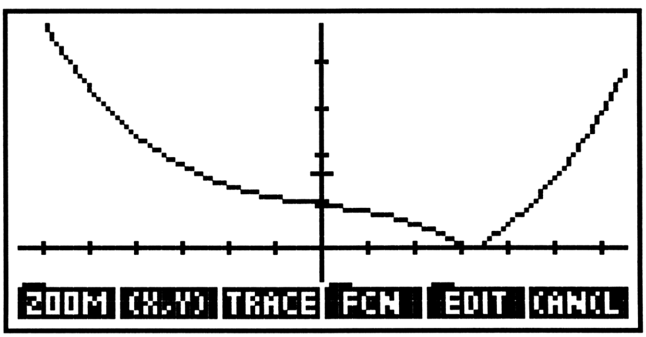
The next few sections describe how to cope with these special kinds of definite integrals, and they provide a set of programmed tools to use for each situation.

Segmenting a Definite Integral

Integrands with cusps in the interval of integration often require many more sample points to achieve a given level of accuracy than do other integrands. The best way to improve the HP 48's built-in numerical integration routine in such cases is to *segment* the integral—divide it into two (or more) integrals at the cusp(s).

Example: Plot $f(x) = \left|x^3 + x - 2\right|$, then use the AREA command in the FCN menu to compute the definite integral of $f(x)$ between the limits –0.8 and 1.6, to six decimal places.

1. Return to the stack (pressing [CANCEL] a few times), set the display to `6 FIX`, open the **PLOT** application and reset the plot parameters: [6][α][α][F][I][X][ENTER][→][PLOT][DEL][▼][ENTER].
2. In the **EQ:** field enter the function: ['][MTH] **VECTR** **ABS** [α][←][X][y^x][3][+][α][←][X][−][2][ENTER].
3. Change **INDEP:** to `x` (lower-case), **H-VIEW** to `-2 2`, **V-VIEW** to `-3 10`, and draw the plot: [α][←][X][ENTER][2][+/−][ENTER][2][ENTER][▶][3][+/−][ENTER][1][0][ENTER] **ERASE** **DRAW**.



4. Press **(X,Y)**, move the cursor to `X: -0.800000`, then [NXT] **FCN** **AREA** to begin the area computation. Press [NXT] **PICT** **(X,Y)**, move to `X: 1.600000`, and [NXT] **FCN** **AREA** to finish the computation. Result: `AREA: 4.240801`

   Note that it takes nearly two minutes to compute the definite integral via the built-in integration routine (used by AREA as well as [→][∫]).

Now, suppose you break the interval of integration into two subintervals by dividing it at the cusp point. Thus,

$$\int_{-0.8}^{1.6} \left|x^3 + x - 2\right| dx \text{ becomes } \int_{-0.8}^{1} \left|x^3 + x - 2\right| dx + \int_{1}^{1.6} \left|x^3 + x - 2\right| dx.$$

Example: Compute the definite integral of $f(x)$ between -0.8 and 1.6 using the segmented version of the integral expression.

1. Return to the stack, enter the integrand (by recalling `EQ`), and make an extra copy expression: [CANCEL][CANCEL][α][α][E][Q][ENTER][ENTER].
2. Compute the first segment of the integral: [•][8][+/–][ENTER][SWAP] [1][ENTER][SWAP]['][α][←][X][ENTER][→][∫][←][→NUM].

   Result: `3.272400`
3. Compute the second segment of the integral: [SWAP][1][ENTER][SWAP] [1][•][6][ENTER][SWAP]['][α][←][X][ENTER][→][∫][←][→NUM].

   Result: `0.968400`
4. Add the segments together: [+]. Result: `4.240800`

   This time it required less than five seconds of computation to get the same answer returned in the previous example after two minutes! Furthermore, the segmented result is more accurate—the exact integral is $\frac{5301}{1250}$ or 4.2408.

The programs `SEGINT` (see page 322) and `NSEGINT` (see page 304), written by William C. Wickes and first published in his book, *HP 48 Insights, Part II: Problem-Solving Resources* (and included here with his permission), automate the process of segmenting a troublesome integral. Both programs take the symbolic, unsegmented integral expression from level 2 and the value of the independent variable at which to segment the integrand from level 1. `SEGINT` returns the symbolic, segmented integral expression, while `NSEGINT` returns the numeric estimate of the segmented integral.

If you desire a numerical estimate, NSEGINT will be quicker but using SEGINT and then pressing [←][→NUM] allows you to check that the segmentation occurred as you expected before using it to estimate the integral.

Example: Use both SEGINT and NSEGINT to segment the integral at $x = -1$, then estimate its value to five places: $\int_{-3}^{3}\sqrt{|x^3+1|}$

1. Set the display mode to 5 FIX: [5][α][α][F][I][X][ENTER]
2. Now enter the unsegmented integral expression: [←][EQUATION][→][∫][+/−][3][▶][3][▶][√x][MTH] VECTR ABS [α][←][X][y^x][3][▶][+][1][▶][▶][▶][α][←][X][ENTER].
3. Enter the point of segmentation: [1][+/−][ENTER]
4. Make a copy of the first two stack levels so that you avoid reentering the arguments: [←][STACK][NXT] DUP2.
5. Use SEGINT to symbolically segment the integral: [α][α][S][E][G][I][N][T][ENTER] or [VAR] (then [NXT] or [←][PREV] as needed) SEGIN.

   Result: '∫(-3,-1,√ABS(x^3+1),x)+
   ∫(-1,3,√ABS(x^3+1),x)'
6. Evaluate the segmented integral: [←][→NUM]. Result: 13.53903
7. Drop the previous result and repeat the computation in a single step using NSEGINT: [◀][α][α][N][S][E][G][I][N][T][ENTER] or [VAR] ([NXT] or [←][PREV] as needed) NSEGI. Result: 13.53903

Improper Integrals: Unbounded Intervals

Normally, definite integrals are defined over a *closed* and *bounded* interval—*bounded* because both limits are finite and *closed* because the integrand is defined at both limits. An *improper integral* is a finite integral over either an unbounded or open interval. That is, an improper integral has a finite value even when it uses an infinite number of approximating rectangles or an infinitely "tall" approximating rectangle. In all cases, the first thing to determine when confronted with an improper integral is whether it *converges* to a finite value or whether it *diverges* to an infinite value. Only those that converge merit further attention.

You may need to use a variety of analytic techniques to discover whether or not an improper integral converges or diverges. The programs SERX1, SERX2, and SERX3, described in Chapter 1 (see page 32), may help—but pay attention to the underlying requirements for the viable use of each test! Remember, too, that the use of these programs alone may not be sufficient to determine convergence.

Example: Determine whether the following improper integral converges:

$$\int_0^{\infty} \frac{1}{4+x^2}\,dx$$

1. Enter the integrand and make a copy: ['] [1] [÷] [↰] [( )] [4] [+] [α] [↰] [X] [y^x] [2] [ENTER] [ENTER].
2. Enter the independent variable and do a root test: ['] [α] [↰] [X] [ENTER] [α] [α] [S] [E] [R] [X] [1] [ENTER]. <u>Result</u>: "Inconclusive"
3. The root test is inconclusive, so drop the result string, make another copy of the integrand, and enter a convergent comparison function, $1/x^2$, a logical choice: [←] [ENTER] ['] [1] [÷] [α] [↰] [X] [y^x] [2] [ENTER].
4. Enter the independent variable and apply the limit comparison test using SERX2: ['] [α] [↰] [X] [ENTER] [α] [α] [S] [E] [R] [X] [2] [ENTER].

   <u>Result</u> (to 5 places): { .96154 0.99960 1 1 1 1 1 }

   The comparison function converges and the limit comparison test converges to a number greater than zero, so you can conclude that the improper integral converges.

Example: Determine whether $\int_e^{\infty} \frac{1}{x \ln x}\, dx$ converges.

1. This integrand is difficult to evaluate for convergence. None of the three tests prove conclusive (check, if you wish), so the next most straightforward trick is to solve the integral symbolically and allow the independent variable approaches zero. The substitution $u = \ln x$ (and thus $du = dx/x$) transforms the integral into $\int_1^{\ln \infty} \frac{1}{u}\, du$.
2. Evaluating it symbolically yields $\int_1^{\ln \infty} \frac{1}{u}\, du = \ln u \Big]_1^{\ln \infty} = \ln(\ln x) \Big]_e^{\infty}$
3. Clearly, as x approaches infinity, the value of the integral approaches infinity—although very slowly. Therefore the improper integral diverges and is thus actually an *impossible integral.*

Another option for determining the convergence of an improper integral is to use the program CDINT (see page 287). Although CDINT shares the inevitable problem of all programs that approximate infinite limiting behavior using finite numerical algorithms—it fails for some situations—it can give you a fairly reliable judgement about the convergence of a particular integral. Unlike SERX3 (the integral test), it doesn't constrain the nature of the integral being tested to those that are continuous, positive, and decreasing.

CDINT takes an integral expression from level 4 that contains a variable name instead of the problematic limit (i.e. infinite values for unbounded intervals or asymptotic values for open intervals, as you'll see in the next section). It takes the name of the substitute variable from level 3, the problematic value of the limit from level 2, and either -1 or 1 from level 1 to indicate whether the limit is being approached from below (-1) or above (1). Note that you may use the infinity character, ∞ ([α][→][I]), to represent infinite limits. CDINT will return either "Diverges", "Converges~*x.xxx*" (where *x.xxx* is an approximation of the value), or "Limit: *ssss*" (where *ssss* may be either a number or an expression that you can evaluate to see if it converges or diverges at the problematic limit).

Example: Use COINT to determine the convergence of $\int_2^\infty \frac{1}{x^2-1}\,dx$.

1. Enter the integral expression, using a variable name (say, *a*) instead of the problematic limit (∞ in this case): ['][→][∫][2][←]['][α][←][A] [←]['][1][÷][←][( )][α][←][X][y^x][2][−][1][▶][←]['][α][←][X][ENTER].
2. Enter the name of the substitute variable: ['][α][←][A][ENTER].
3. Enter the value of the problematic limit: ['][α][→][I][ENTER].
4. Enter -1 (the limit is approached from below): [1][+/−][ENTER].
5. Determine convergence using COINT: [α][α][C][D][I][N][T][ENTER] or [VAR] ([NXT] or [←][PREV]) COINT. Result: "Converges~0.549"

A couple of notes: The value returned by COINT when the integral converges is approximate—don't use it blindly as the value of the integral itself. In this particular case, the integral reduces analytically to 0.5(ln 3) ≈ 0.549306144335 and COINT agrees with it to six places—an unusually good agreement. COINT may take a long time to determine the convergence of some integrals. Use it carefully, as one of your tools—not the only tool—for determining convergence.

Once you have confirmed that a particular integral with infinite limits converges to a finite value, you can accurately estimate that value by mapping the infinite interval to a finite interval via a *change of the variable of integration.* Suppose, for example, that $\int_a^\infty f(x)\,dx$ converges. Then, by defining a variable $u=\tan^{-1}(x)$ (or $x = \tan u$), you can rewrite the improper integral as $\int_{\tan^{-1} a}^{\tan^{-1}\infty} f(\tan u)\left(\frac{d(\tan u)}{du}\right) du$.

Notice that the new limits are now finite: $\tan^{-1} a$ and $\pi/2$. This particular change of variables maps the entire real *x*-axis onto the finite interval $-\pi/2 \le u \le \pi/2$. Estimating the definite integral using this transformed version should be quicker and more accurate and isn't going to be subject to as much round-off error.

Two programs are provided to assist you in making the change of variables and using the transformed integral to estimate the numerical integral.

The first, CHVAR (page 288), based on the program of the same name written by William C. Wickes in his *HP 48 Insights, Vol. II: Problem-Solving Resources*, takes an integral expression from level 2 and an equation defining the new variable from level 1 and returns the transformed expression. The defining equation should be of the form ' $var_{new} = g(var_{old})$ ' , where var_{new} is the new variable of integration, var_{old} is the old variable of integration, and g the function that relates them.*

The second, UBINT (page 332), uses CHVAR and the $u = \tan^{-1}(x)$ transformation to compute the numerical integral for an integral expression with an unbounded interval—*providing that the integral converges to a finite limit.* UBINT takes the integral expression from level 1 and returns the estimate to level 1. Note that you may either use the built-in constant, 'MAXR' (9.9999999999E499) as a stand-in for infinity or the '∞' ([α][→][1]) character.

The following examples illustrate the use of CHVAR and UBINT.

Example: Use CHVAR to transform $\int_0^1 2x(x^2-1)^4\,dx$ to a simpler one, by using the transformation $u = x^2 - 1$.

1. Enter the integral expression: ['][→][∫][0][←]['][1][←][,][2][×][α][←][X][×][←][()][α][←][X][yˣ][2][−][1][▶][yˣ][4][←][,][α][←][X][ENTER].
2. Enter the transformation equation: ['][α][←][U][←][=][α][←][X][yˣ][2][−][1][ENTER].
3. Perform the change-of-variables transformation: [α][α][C][H][V][A][R][ENTER] or [VAR] (then [NXT] or [←][PREV] as needed) **CHVAR**.

   Result: '∫(-1,0,u^4,u)'

*Note that CHVAR can be used for any change-of-variable situation—not merely for resolving improper integrals.

Example: Use UBINT to estimate the value of the following unbounded integral (which you've already shown to be convergent):

$$\int_0^\infty \frac{1}{4+x^2}\,dx$$

1. Enter the integral expression: ['] [→] [∫] [0] [←] [,] [α] [→] [I] [←] [,] [1] [÷] [←] [( )] [4] [+] [α] [←] [X] [y^x] [2] [▶] [←] [,] [α] [←] [X] [ENTER].
2. Set the display to STD format and compute the improper integral: [α] [α] [S] [T] [D] [ENTER] [α] [α] [U] [B] [I] [N] [T] [ENTER] or [VAR] ([NXT] or [←] [PREV]) [UBINT]. Result: .785398163393

   This compares well with the analytical result: $\pi/4 \approx .785398163398$.

Remember that UBINT should not be used on improper integrals for which you haven't yet determined convergence.

Example: Use UBINT to estimate the value of $\int_e^\infty \frac{1}{x \ln x}\,dx$ (which you have already shown to be divergent).

1. Enter the integral expression: ['] [→] [∫] [α] [←] [E] [←] [,] [α] [→] [I] [←] [,] [1] [÷] [←] [( )] [α] [←] [X] [×] [→] [LN] [α] [←] [X] [▶] [▶] [←] [,] [α] [←] [X] [ENTER].
2. Compute the improper integral using UBINT: [UBINT]

   Result (to 3 places): 2.883 (after a long wait)

   Furthermore, the result is misleading since the integral diverges. Remember to check for convergence first before using UBINT.

Improper Integrals: Open Intervals

Integrals that have unbounded intervals are improper because they require an infinite number of approximating rectangles. But integrals whose intervals contain a vertical *asymptote* are also "improper" because they require an approximating rectangle of infinite "height."

For this second kind of improper integral, you must again first determine whether or not it converges to a finite value before attempting to estimate that value.

Example: Determine whether $\int_1^2 (x-2)^{-\frac{2}{3}}\,dx$ converges.

1. Enter the integral expression with the problematic limit (2) replaced by a variable (say, b): ['] [↱] [∫] [1] [↰] [,] [α] [↰] [B] [↰] [,] [↰] [( )] [α] [↰] [X] [−] [2] [▶] [yˣ] [↰] [( )] [+/−] [2] [÷] [3] [▶] [↰] [,] [α] [↰] [X] [ENTER].
2. Enter the limit variable and its value: ['] [α] [↰] [B] [ENTER] [2] [ENTER].
3. Enter -1 (approach limit from below) and use `CDINT`: [1] [+/−] [ENTER] [α] [α] [C] [D] [I] [N] [T] [ENTER]. Result: `Limit: 3` It converges near 3.

The program `OPINT` (page 305) estimates the value of an integral that has a vertical asymptote at one of its endpoints—i.e. an *open interval.* It uses a modified version of the built-in integration algorithm designed so that the precise endpoints are not used in the estimation process. `OPINT` takes the improper integral expression from level 1 and returns a numerical estimate to level 1.

Example: Estimate $\int_1^2 (x-2)^{-\frac{2}{3}}\,dx$ (which you previously found convergent).

1. Enter the integral expression: ['] [↱] [∫] [1] [↰] [,] [2] [↰] [,] [↰] [( )] [α] [↰] [X] [−] [2] [▶] [yˣ] [↰] [( )] [+/−] [2] [÷] [3] [▶] [↰] [,] [α] [↰] [X] [ENTER].
2. Compute the improper integral: [α] [α] [O] [P] [I] [N] [T] [ENTER] or [VAR] (then [NXT] or [↰] [PREV] as needed) `OPINT`.

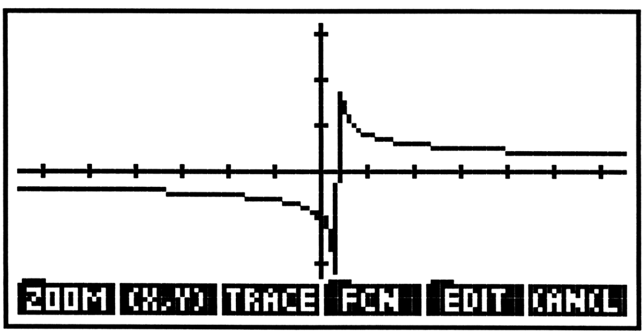
   Result: `3` The limit found by `CDINT` was exact!

When the asymptote falls *within* the interval of integration (i.e. not on either endpoint), a potential problem arises. The interval of integration is actually *two open intervals*—one "below" the asymptote and one "above" it. For the overall integral to converge, *both subintervals* must converge. Usually—but not always—both either diverge or converge. When both subintervals converge, `∂PINT` will usually provide an accurate estimate of the overall integral without requiring you to segment it first. However, it can fail if it should happen to pick the asymptote exactly as a test point during its algorithm. To avoid this, segment the integral at the asymptote and then use `∂PINT` to estimate each segment.

Clearly, the safest approach is to segment the integral at the asymptote and treat each subinterval separately—testing it for convergence, then using `∂PINT` to estimate its value—and, as the last step, total the two estimates. The following examples illustrate both the safe and unsafe (though speedy) approaches.

Example: Estimate the following improper integral by plotting the integrand to determine the location of any asymptotes, segmenting it at the asymptote, making sure that both segments converge, and then using `∂PINT` to find the value of each segment: $\displaystyle\int_0^3 \frac{1}{\sqrt[3]{3x-1}}\,dx$

1. Open the **PLOT** application, set **TYPE:** to `Function`, and reset the plot parameters: [→][PLOT][▲][α][F][DEL][▼][ENTER].
2. Enter the integrand into the **EQ:** field: [▼]['][1][÷][→][ˣ√y][3][←][,][3][×][α][←][X][−][1][ENTER].
3. Set **INDEP:** to `x` and draw the plot: [α][←][X][ENTER] **ERASE** **DRAW**.



4. Pressing TRACE (X,Y) and moving across the asymptote shows that the asymptote is between 0.3 and 0.4. A quick inspection of the integrand suggests that $x = 1/3$ (0.333333) is the asymptote.

5. Return to the stack and enter the integral expression. Note that for best accuracy, use the exponential form for roots: CANCEL CANCEL ' → ∫ 0 ← , 3 ← , ← ( ) 3 × α ← X − 1 ▶ y^x ← ( ) +/− 1 ÷ 3 ▶ ← , α ← X ENTER.

6. Use SEGINT to segment the integral at 1/3: 3 1/x α α S E G I N T ENTER.

7. Separate the two segments and make an extra copy of each: PRG LIST OBJ→ ← STACK NXT DROP2 DUP2.

8. Determine the convergence of the first segment: SWAP ← EDIT ▶ ▶ ▶ ▶ ▶ DEL→ α ← B ← , ENTER ' α ← B ENTER 3 1/x 1 +/− ENTER α α C D I N T ENTER.

   Result: "Limit: -.499999995"

9. Determine the convergence of the second segment: ◀ ← EDIT ▶ ▶ ▶ DEL→ α ← A ← , 3 ← , ENTER ' α ← A ENTER 3 1/x 1 ENTER α α C D I N T ENTER.

   Result: "Limit: 1.999999995"

10. Since both segments converge, compute the value of each segment with OPINT and sum the total: ◀ α α O P I N T ENTER SWAP α α O P I N T ENTER +.

   Result: 1.5 The exact analytical result, as well.

Example: Estimate $\int_0^3 \frac{1}{\sqrt[3]{3x-1}}\,dx$ directly using OPINT.

1. Enter the integral expression, using the exponential form of the root to improve accuracy: ' → ∫ 0 ← , 3 ← , ← ( ) 3 × α ← X − 1 ▶ y^x ← ( ) +/− 1 ÷ 3 ▶ ← , α ← X ENTER.

2. Find the integral via OPINT: α α O P I N T ENTER.

   Result: 1.5

Plotting Antiderivatives

In the previous sections, you've seen how the HP 48 handles situations where the desired result is a real number—representative of the finite area under a specific curve between two given limits. But the object of an integration may instead be a specific function or family of functions. This requires the symbolic capabilities of the HP 48. Consider this:

Treating the integral of a function ($f(x)$) as a function itself ($g(x)$), yields

$$g(x) = \int f(x)\,dx \text{ and thus, } g'(x) = f(x).$$

That is, g is the integral (or *antiderivative*) of f and f is the derivative of g. Notice that g represents a family of functions because there are any number of functions, $g(x)$, whose derivative is $f(x)$. For example, if $f(x) = 3x^2 + 2$, then any of the following are possible antiderivatives:

$$\begin{aligned} g(x) &= x^3 + 2x - 1 \\ g(x) &= x^3 + 2x + 4 \\ g(x) &= x^3 + 2x - 9 \\ g(x) &= x^3 + 2x + 17 \end{aligned}$$

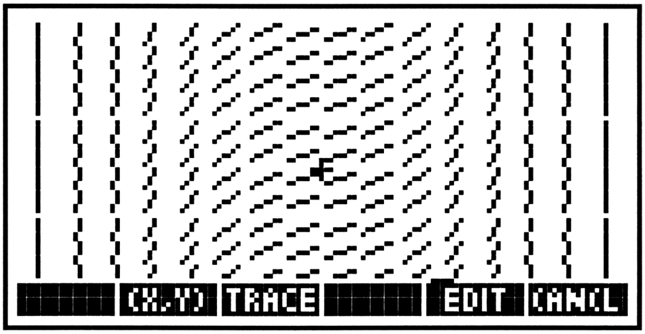
Indeed, the general antiderivative here—also known as the *indefinite integral*—is $x^3 + 2x + C$, where C is any real number.

Believe it or not, the HP 48 can simulate a plot of the indefinite integral! The Slopefield plot type plots the equation, $g'(x) = f(x)$, by using a set of x-values and computing the *slope* of the antiderivative g at each point. It then draws a short line segment with the computed slope at a given set of points on the y-axis. The Slopefield plot allows you to visualize from short slope segments the actual set of functions (i.e. the indefinite integral) for which $f(x)$ is a derivative.

Example: Draw a slopefield for $f(x) = 3x^2 + 2 = g'(x)$. Use 16 x-values and plot the computed slope at 12 different spots along the vertical axis.

1. Open the PLOT application and set the TYPE: to Slopefield: [→][PLOT][▲][α][S][α][S].

2. Enter the function in EQ: : [▼] ['] [3] [×] [α] [←] [X] [y^x] [2] [+] [2] [ENTER].
3. Set INDEP: to x (lowercase), STEPS: (INDEP) to 16, and STEPS: (DEPND) to 12: [α] [←] [X] [ENTER] [1] [6] [ENTER] [▶] [1] [2] [ENTER].
4. Now set X-LEFT to −4, X-RIGHT to 4, then Y-NEAR to −15, and Y-FAR to 15: OPTS [4] [+/−] [ENTER] [4] [ENTER] [1] [5] [+/−] [ENTER] [1] [5] [ENTER] [ENTER].
5. Draw the slopefield plot: ERASE DRAW.



You can see the general shape of a cubic polynomial antiderivative swept out by the slope segments in this plot.

The indefinite integral is a family of functions, as the slopefield plot illustrates. To specify which particular member of the family solves a given problem, remember that the family of functions each differ from one another by the value of the constant, C. If you can specify C, you can specify the function you need. You can find C if you know the value of the antiderivative function at at least one point. This point is usually referred to as the *initial condition* because it requires (i.e. "conditions") the function to pass through it.

Returning to the equation in the previous example ($g'(x)=3x^2 + 2$), suppose you also knew an initial condition, say that $g(0) = 3$. You can now specify C and thus a specific antiderivative:

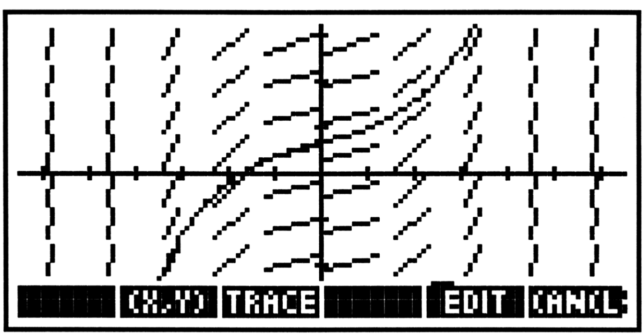
$$\begin{aligned} &g(x) = x^3 + 2x + C \\ &g(0) = (0)^3 + 2(0) + C = 3 \\ &C = 3 \\ &\underline{\underline{g(x) = x^3 + 2x + 3}} \end{aligned}$$

Thus, the function $x^3 + 2x + 3$ passes through the point (0,3), the initial condition.

The program ANTI∂ (see page 286) plots the specific antiderivative function given initial conditions on a general slopefield plot. It uses the function, $f(x)$, on level 6, a list of the independent and dependent variables on level 5, the plotting range (as a list) on level 4, the horizontal display range (as a list) on level 3, the vertical display range (as a list) on level 2, and the initial condition point(s) (as a complex number—or list of complex numbers to find more than one antiderivative) on level 1. ANTI∂ uses EULPLT (page 293), which finds antiderivatives from initial conditions using the quick (but low-precision*) Euler algorithm.

Example: Use ANTI∂ to plot the slopefield of the equation $g'(x) = 3x^2 + 2$ and the specific antiderivative corresponding to the initial condition: $g(0) = 3$. Use a plotting range of $-3 \le x \le 3$, a horizontal display range of $-4 \le x \le 4$, and a vertical display range of $-15 \le y \le 15$.

1. The function: [CANCEL] [CANCEL] ['] [3] [×] [α] [←] [X] [y^x] [2] [+] [2] [ENTER].
2. The list of independent and dependent variables: [←] [{ }] [α] [←] [X] [SPC] [α] [←] [Y] [ENTER].
3. The plotting range (as a list): [←] [{ }] [3] [+/−] [SPC] [3] [ENTER].
4. The horizontal display range (as a list): [←] [{ }] [4] [+/−] [SPC] [4] [ENTER].
5. Enter the vertical display range: [←] [{ }] [1] [5] [+/−] [SPC] [1] [5] [ENTER].
6. The initial condition (as a complex no.): [←] [( )] [0] [←] [,] [3] [ENTER].
7. Plot the slopefield and the specific antiderivative using ANTI∂: [α] [α] [A] [N] [T] [I] [→] [D] [ENTER] or [VAR] ([NXT] or [←] [PREV]) [ANTI∂].



*For most situations, the Euler algorithm generates an adequate plot, but shouldn't be relied upon for computed solutions. Instead, use the built -in differential equation solvers, RKF and RRK, that are based upon the fourth-order Runge-Kutta-Fehlberg method. These are not discussed in this book.

Indefinite Integrals: Symbolic Antiderivatives

Finding the indefinite integral—the general antiderivative— of a function is a common task in integral calculus. Although you can numerically approximate the integral for a specified interval or use plotting techniques to give you a visual approximation of either the general or specific antiderivative, sometimes what you need most is the general antiderivative in symbolic form.

As mentioned on page 132, the HP 48's built-in capabilities for finding symbolic antiderivatives are limited to matching patterns from a built-in list (see pages 133-134). As long as the integral in question can be dealt with using the built-in list, it's quite easy to use the integration command—which requires upper and lower limits—to actually perform an indefinite integration.

Example: Use the built-in integration command to find $\int (3x^2 + p)\,dx$, where p is an unspecified constant.

1. At the stack, purge the independent variable and formal constant, p: [CANCEL] [CANCEL] [←] [{ }] [α] [←] [X] [SPC] [α] [←] [P] [ENTER] [←] [PURG].
2. Enter the integral expression, using any number for the lower limit and the independent variable name for the upper limit: ['] [→] [∫] [1] [←] [,] [α] [←] [X] [,] [3] [×] [α] [←] [X] [y^x] [2] [+] [α] [←] [P] [←] [,] [α] [←] [X].
3. Evaluate the integral: [EVAL]. The HP 48 finds a match for the integral and returns the evaluated integral before substituting the limits.
4. Drop the lower limit part of the resulting expression and finish evaluating the integral: [PRG] [LIST] [OBJ→] [◆] [◆] [◆] [EVAL].

   <u>Result</u>: `'p*x+3*(x^3/3)'`

The program `INDEF` (see page 296) helps automate the steps required to compute an indefinite integral. It doesn't require that the variable of integration be purged first. `INDEF` can use either of two sets of arguments. The first option takes the integrand expression from level 2, and the variable of integration from level 1. The second option takes only an integral expression from level 1. If successful at finding a complete antiderivative, `INDEF` returns it to level 2 and a `1` to level 1, indicating success. If INDEF can't find a complete antiderivative, it returns either an unevaluated integral expression or a partial antiderivative to level 2 (using dummy limits) and a 0 to level 1, indicating failure. INDEF is used by several other programs in this book.

Example: Use INDEF to find $\int (3x^2 + p)\,dx$.

1. Enter the integrand: ['] [3] [×] [α] [⬅] [X] [y^x] [2] [+] [α] [⬅] [P] [ENTER].
2. Enter the variable of integration: ['] [α] [⬅] [X] [ENTER].
3. Now compute the indefinite integral with INDEF: [α] [α] [I] [N] [D] [E] [F] [ENTER] or [VAR] (then [NXT] or [⬅] [PREV] as needed) [INDEF].

   Result:
   ```
   2: 'p*x+3*(x^3/3)'
   1:                    1
   ```
   —a successful match.

Example: Use INDEF to find $\int \frac{1}{4+x^2}\,dx$.

1. Enter the integrand: ['] [1] [÷] [⬅] [( )] [4] [+] [α] [⬅] [X] [y^x] [2] [ENTER].
2. Enter the variable of integration and compute the indefinite integral: ['] [α] [⬅] [X] [ENTER] [α] [α] [I] [N] [D] [E] [F] [ENTER].

   Result:
   ```
   2: '∫(1,x,1/(4+x^2),x)'
   1:                    0
   ```

   INDEF fails and returns an unevaluated integral expression instead.

Example: Use INDEF to find $\int x^2 + \frac{1}{4+x^2}\,dx$.

1. Enter the integral expression, using arbitrary limits, and compute the indefinite integral: ['] [↱] [∫] [0] [⬅] [,] [1] [⬅] [,] [α] [⬅] [X] [y^x] [2] [+] [1] [÷] [⬅] [( )] [4] [+] [α] [⬅] [X] [y^x] [2] [▶] [⬅] [,] [α] [⬅] [X] [ENTER] [INDEF].

   Result:
   ```
   2: 'x^3/3+∫(1,x,1/(4+x^2),x)'
   1:                          0
   ```

   INDEF finds part of the antiderivative and returns the mixed result.

Expanding the Pattern-Matching Capability

It won't take you long to find integrands for which the HP 48 can't find a pattern-match in its built-in list. But you can add other patterns by creating an additional list that the HP 48 will search if it fails to find a match in the built-in list.

The approach used in this book to building a supplemental list of integration patterns is borrowed from William C. Wickes and his book *HP 48 Insights, Part II: Problem-Solving Resources.* His 3 programs, XINT, LINEAR?, and FUNOF?, are included here with his permission (though LINEAR? is named LININ? here).

XINT (see page 334) is a replacement for the built-in integration command. It searches a supplement list of patterns, named IPATS, if the built-in command fails to find a match. LININ? (page 300) and FUNOF? (page 294) are user-defined functions that augment the flexibility of the patterns included in IPATS just as the patterns in the built-in list have been. They are used behind the scene by ADDPAT as it builds entries for IPATS.

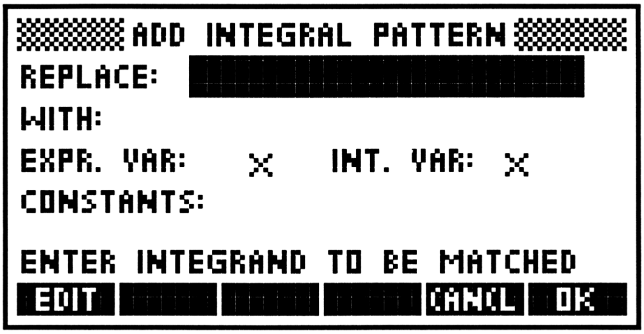
The program ADDPAT (see page 285) has been added to Wickes's capable collection to provide a more user-friendly means of adding patterns to IPATS. It takes nothing from the stack and returns nothing. Instead, it uses an input screen to prompt for information about the pattern you're adding, then processes the information and adds an entry to IPATS in the proper syntax. Note that IPATS doesn't have to exist before using ADDPAT.

Example: Use ADDPAT to add the following antiderivative patterns to IPATS:

$$\int \frac{1}{a^2 + x^2} dx = \frac{1}{a} \tan^{-1} \frac{x}{a} + C$$

$$\int x e^x dx = (x - 1)e^x + C$$

1. Begin ADDPAT: (α)(α)(A)(D)(D)(P)(A)(T)(ENTER) or (VAR) (then (NXT) or (←)(PREV) as needed) ADDPA....



2. Enter the integrand pattern in the REPLACE: field: ['] [1] [÷] [←] [()] [α] [←] [A] [y^x] [2] [+] [α] [←] [X] [y^x] [2] [ENTER].
3. Enter the antiderivative pattern to be substituted for the integrand pattern: ['] [1] [÷] [α] [←] [A] [×] [ATAN] [α] [←] [X] [÷] [α] [←] [A] [ENTER].
4. The variable used in the patterns and the variable of integration is the same—X, the default. Skip over the third line and enter the list of constants used in the REPLACE: pattern in the CONSTANTS: field: [▼] [←] [{ }] [α] [←] [A] [ENTER].
5. Press [OK] to add the pattern to IPATS.
6. Repeat steps 2 through 5 for the second pattern. Note that the list of constants is empty: ['] [α] [←] [X] [×] [←] [e^x] [α] [←] [X] [ENTER] ['] [←] [()] [α] [←] [X] [−] [1] [►] [×] [←] [e^x] [α] [←] [X] [ENTER] [▼] [←] [{ }] [ENTER] [OK].
7. Return to the stack and inspect IPATS: [CANCEL] [α] [α] [I] [P] [A] [T] [S] [ENTER]. The two newest entries begin the list:

```
{ { '&1*EXP(&1)' '(&1-1)*EXP(&1)'
    'LININ?(QUOTE(&1),QUOTE(↑↓))' }
  { '1/(&2^2+&1^2)' '1/&2*ATAN(&1/&2)' 'NOT
    FUNOF?(QUOTE(&2),↑↓) AND LININ?(QUOTE(&1),
    QUOTE(↑↓))' } ... }
```

Notice the inclusion of LININ? and FUNOF? tests in the IPATS list. The LININ? test allows ∫INT to find a pattern that uses a linear function of the independent variable instead of just the variable itself. FUNOF? assures that the constants are truly constant and not functions of the independent variable—that is, for example, that the constant b in an expression doesn't evaluate to, say, $2x^2 + 7$.

Example: Use ADDPAT to add the following antiderivative patterns to IPATS:

$$\int \frac{c}{x(x+d)}\,dx = c\ln\left(\frac{x}{x+d}\right) + C$$

$$\int \sin ax \cos ax\, dx = -\frac{\cos 2ax}{4a} + C$$

$$\int \frac{b}{x \ln ax}\,dx = b\ln\left|\ln ax\right| + C$$

1. Begin ADDPAT again and add the first pattern: [α] [α] [A] [D] [D] [P] [A] [T] [ENTER] ['] [α] [←] [C] [÷] [←] [( )] [α] [←] [X] [×] [←] [( )] [α] [←] [X] [+] [α] [←] [D] [ENTER] ['] [α] [←] [C] [×] [→] [LN] [α] [←] [X] [÷] [←] [( )] [α] [←] [X] [+] [α] [←] [D] [ENTER] [▼] [←] [{ }] [α] [←] [C] [SPC] [α] [←] [D] [ENTER] [OK].
2. Add the second pattern: ['] [SIN] [α] [←] [A] [×] [α] [←] [X] [▶] [×] [COS] [α] [←] [A] [×] [α] [←] [X] [ENTER] ['] [+/−] [COS] [2] [×] [α] [←] [A] [×] [α] [←] [X] [▶] [÷] [←] [( )] [4] [×] [α] [←] [A] [ENTER] [▼] [←] [{ }] [α] [←] [A] [ENTER] [OK].
3. Add the third pattern: ['] [α] [←] [B] [÷] [←] [( )] [α] [←] [X] [×] [→] [LN] [α] [←] [A] [×] [α] [←] [X] [ENTER] ['] [α] [←] [B] [×] [→] [LN] [MTH] [VECTR] [ABS] [→] [LN] [α] [←] [A] [×] [α] [←] [X] [ENTER] [▼] [←] [{ }] [α] [←] [A] [SPC] [α] [←] [B] [ENTER] [OK].
4. Return to the stack: [CANCEL].

Caveat: Before you make a large IPATS list and use it routinely, bear in mind that IPATS uses available memory and that the speed with which integrals are evaluated is noticeably lowered if large IPATS lists must be searched.

Use XINT wherever you would normally use the built-in [→][∫]. XINT takes the same four arguments, in the same order, as ∫: the lower limit from level 4, the upper limit from level 3, the integrand from level 2, and the variable of integration from level 1. If XINT finds a match, it returns the partially evaluated integral; if it fails to find a match, it returns the unevaluated integral expression.

Example: Use XINT to evaluate $\int_a^b \frac{1}{4+x^2}\,dx$

1. Enter the limits of integration: ['][α][←][A][ENTER]['][α][←][B][ENTER].
2. Enter the integrand and variable of integration: ['][1][÷][←][( )][4][+][α][←][X][y^x][2][ENTER]['][α][←][X][ENTER].
3. Evaluate the integral with XINT: [α][α][X][I][N][T][ENTER] or [VAR] ([NXT] or [←][PREV]) XINT. Result: '∫(a,b,1/(4+x^2),x)'

XINT failed to find a match—even though IPATS contains the relevant pattern (the first one you entered). Why? Because the pattern-matching is looking for a^2 and doesn't recognize 4 as 2^2. To get XINT to do this integral, you must use 2^2 or add a modified pattern to IPATS: $\int \frac{1}{a+x^2}\,dx = \frac{1}{\sqrt{a}}\tan^{-1}\frac{x}{\sqrt{a}} + C$

Example: Add the new pattern to IPATS and then repeat the previous example.

1. Enter the new pattern: [α][α][A][D][D][P][A][T][ENTER]['][1][÷][←][( )][α][←][A][+][α][←][X][y^x][2][ENTER]['][1][÷][$\sqrt{x}$][α][←][A][×][←][ATAN][α][←][X][÷][$\sqrt{x}$][α][←][A][ENTER][▼][←][{ }][α][←][A][ENTER] OK.
2. Return to the stack and enter the limits of integration: [CANCEL]['][α][←][A][ENTER]['][α][←][B][ENTER].
3. Enter the integrand and variable of integration: ['][1][÷][←][( )][4][+][α][←][X][y^x][2][ENTER]['][α][←][X][ENTER].
4. Evaluate: [α][α][X][I][N][T][ENTER] or [VAR] ([NXT] or [←][PREV]) XINT.

   Result: '1/√4*ATAN(↑↓/√4)|(↑↓=b)-(1/√4*ATAN(↑↓/√4)|(↑↓=a)'
5. Now XINT finds a match. Press [EVAL] to complete the evaluation of the integral. Result: '.5*ATAN(b/2)-.5*ATAN(a/2)'

Example: Use XINT to evaluate $\displaystyle\int_a^b \frac{2}{(3x-1)(3x+7)}\,dx$.

1. Analyze the situation. This integral doesn't obviously match any of the integration patterns you added to IPATS. However, notice that the second factor in the denominator can be re-expressed as (3*x*-1+8). If you use that form with XINT, it should match the third pattern you entered—with *x* represented by its linear function 3*x* – 1.
2. Enter the limits of integration: ['] [α] [←] [A] [ENTER] ['] [α] [←] [B] [ENTER].
3. Enter the modified integrand and integration variable: [←] [EQUATION] [2] [÷] [←] [( )] [3] [α] [←] [X] [−] [1] [▶] [←] [( )] [3] [α] [←] [X] [−] [1] [+] [8] [ENTER] ['] [α] [←] [X] [ENTER].
4. Evaluate the integral with XINT: [α] [α] [X] [I] [N] [T] [ENTER] [EVAL].

   Result: `'2*LN((3*b-1)/(3*b-1+8))-2*LN((3*a-1)/(3*a-1+8)'`

Example: Use XINT to evaluate $\displaystyle\int_a^b \sin(x+2)\cos(x+2)\,dx$.

1. Enter the limits of integration: ['] [α] [←] [A] [ENTER] ['] [α] [←] [B] [ENTER].
2. Enter the integrand and variable of integration: ['] [SIN] [α] [←] [X] [+] [2] [▶] [X] [COS] [α] [←] [X] [+] [2] [ENTER] ['] [α] [←] [X] [ENTER].
3. Evaluate the integral with XINT: [α] [α] [X] [I] [N] [T] [ENTER].

   Result: `'∫(a,b,SIN(x+2)*COS(x+2),x)'`

   XINT fails because the relevant pattern contains `a*x` while this integral has only an *implied* 1 for *a*. Either explicitly include a 1 in the integrand—`SIN(1*(x+2))*COS(1*(x+2))`—or add a pattern to IPATS that expresses the pattern without *a*:

$$\int \sin x \cos x\,dx = -\frac{\cos 2x}{4} + C$$

Example: Use XINT to evaluate $\displaystyle\int_a^b \frac{3}{(4x+3)\ln[5(4x+3)]}\,dx$.

1. Enter the limits of integration: ['][α][←][A][ENTER]['][α][←][B][ENTER].
2. Enter the integrand and variable of integration: ['][3][÷][←][( )][←][( )][4][×][α][←][X][+][3][▶][×][→][LN][5] [×][←][( )][4][×][α][←][X][+][3][ENTER]['][α][←][X][ENTER].
3. Evaluate the integral with XINT: [α][α][X][I][N][T][ENTER][EVAL].

   Result: '3*LN(ABS(LN(5*(4*b+3))))-3*
   LN(ABS(LN(5*(4*a+3))))'

Because XINT is an exact replacement for the built-in integral command (∫), it can be substituted for ∫ in any program to extend the program's pattern-matching capabilities to the supplemental IPATS list. The program XINDEF (see page 333) is included here as an example. It differs from INDEF only in that the ∫ command has been replaced by XINT and INDEF by XINDEF.

Example: Use XINDEF to compute $\displaystyle\int \sin 3x \cos 3x\,dx$.

1. Enter the integrand: ['][SIN][3][×][α][←][X][▶][×][COS][3][×][α][←][X][ENTER].
2. Enter the variable of integration: ['][α][←][X][ENTER].
3. Compute the indefinite integral using XINDEF: [α][α][X][I][N][D][E][F][ENTER] or [VAR] (then [NXT] or [←][PREV] as needed) XINDE.

   Result: 2: '-(COS(6*x)/12'
   1: 1

You can continue this process of extending the symbolic capabilities to other programs, replacing INDEF with XINDEF in programs such as OPINT, UBINT, CDINT, or IPARTS; these programs will then search IPATS as well as the built-in list for matches for integrands. Keep in mind, of course, that extending the search may also slow down the performance of the programs.

Modifying the Function being Integrated

As you were trying to produce integrands that the steadfast-but-not-too-bright pattern-matcher used by the HP 48 could recognize, no doubt it occurred to you one of the most important tricks to successful symbolic integration is modifying the integrand so that it's easier to use.

There are several kinds of modifications that are quite useful:

- Change of Variables: Making an easier integrand by substituting one variable for another.
- Integration by Parts: Simplifying an integrand by dividing it into more convenient parts.
- Partial Fractions: Converting a complicated rational fraction integrand into a series of simpler rational fractions that are easily integrated.

The next three sections illustrate these methods of simplifying integrands.

Substitution: Change of Variables

Finding the antiderivative of a function is a matter of recognizing that the function is a derivative of some other function. Of course, derivatives are often formed via the chain rule—as the result of a composition of functions. The *method of substitution* (also known as the *change-of-variables method*) is a technique that allows you to recognize derivatives produced by the chain rule and thus to find their antiderivatives. It is a kind of "chain-rule for integration" that asks that you look at an integrand as potentially containing a composition of two functions.

Notice that the derivative of a composite function yields an expression containing both the "inside" function (g) and its derivative (g'): $\frac{d}{dx}f(g(x)) = f'(g(x))g'(x)$.

Thus, whenever you spot an "inside" function—such as one raised to a power, one in the denominator of a rational fraction, or one used as an argument of a transcendental function—you have a potential candidate for the method of substitution.

The method of substitution works as follows:

1. Choose a candidate for the "inside" function and set it equal to a new variable, i.e. let $u = g(x)$.
2. Compute the derivative of the "inside" function, i.e. find $du = g'(x)\,dx$.
3. Substitute u for $g(x)$ and $\dfrac{du}{g'(x)}$ for dx in the original integrand.
4. Algebraically rearrange the integrand so that the original variable (x or dx) doesn't appear, only the new variable (u or du). Note that you may be able to replace remaining appearances of x with their equivalents in terms of u, using the original relationship from step 1. If you are unable to do this, go back to the beginning and try a different candidate function for u.
5. Integrate the new integrand using u as the variable of integration.
6. Substitute $g(x)$ for u in the evaluated integral and simplify, if necessary.
7. Check the solution by differentiating it and comparing it to the original integrand.

The easiest way to use the method of substitution to find a symbolic antiderivative on your HP 48 is to use `CHVAR` and `INDEF`, described in earlier sections (pages 165 and 132, respectively). `CHVAR` creates the modified integral; `INDEF` finds its antiderivative. Of course, in order to be successful, the modified integral must match one of the built-in integral patterns (or, if you use `XINDEF` instead, one of the integral patterns stored in the `IPATS` list).

Example: Use the method of substitution to find $\displaystyle\int \frac{x^2}{x^3+5}\,dx$.

1. Enter the integral expression, using dummy limits: ['] [→] [∫] [α] [←] [A] [←] [,] [α] [←] [B] [←] [,] [α] [←] [X] [y^x] [2] [÷] [←] [( )] [α] [←] [X] [y^x] [3] [+] [5] [▶] [←] [,] [α] [←] [X] [ENTER].
2. Choose a transformation equation. The denominator factor, $x^3 + 5$, seems a good candidate because its derivative is the same order as the remaining factors. Enter the transformation equation: ['] [α] [←] [U] [←] [=] [α] [←] [X] [y^x] [3] [+] [5] [ENTER].

3. Perform the substitution using `CHVAR`: [α][α][C][H][V][A][R][ENTER].

   Result: `'∫(5+a^3,5+b^3,.333333333333/u,u)'`

4. The integrand has been greatly simplified. At this point, you can either use `INDEF` (for symbolic integrations) or [←][→NUM] (for numeric integrations). Since you require symbolic results in this case, execute `INDEF`: [α][α][I][N][D][E][F][ENTER][◄].

   Result: `'.333333333333*LN(u)'`

5. Re-substitute the original expression: [←][{ }][α][←][U][SPC]['][α][←][X][y^x][3][+][5][ENTER][←][SYMBOLIC][NXT] [↑MAT][◄].

   Result: `'.333333333333*LN(x^3+5)'`

6. Check by differentiating: ['][α][←][X][ENTER][α][α][F][D][E][R][ENTER].

   Result: `'.333333333333*(3*x^2/(x^3+5))'`

   This simplifies to the original integrand, thus confirming the result.

Example: Use the method of substitution to find $\int x\sqrt{a^2+b^2x^2}\,dx$.

1. Enter the integral with dummy limits: ['][→][∫][α][←][C][←][,][α][←][D][←][,][α][←][X][×][√x][←][( )][α][←][A][y^x][2][+][α][←][B][y^x][2][×][α][←][X][y^x][2][►][←][,] [α][←][X][ENTER].

2. Choose a transformation equation: $u = a^2 + b^2x^2$. Enter it: ['][α][←][U][←][=][α][←][A][y^x][2][+][α][←][B][y^x][2][×][α][←][X][y^x][2][ENTER].

3. Perform the substitution using `CHVAR`: [α][α][C][H][V][A][R][ENTER].

   Result: `'∫(b^2*c^2+a^2,b^2*d^2+a^2,`
   `.5*√u*b^-2,u)'`

4. Execute `INDEF`: [α][α][I][N][D][E][F][ENTER][◄].

5. Re-substitute: [←][{ }][α][←][U][SPC]['][α][←][A][y^x][2][+][α][←][B][y^x][2][×][α][←][X][y^x][2][ENTER][←][SYMBOLIC][NXT][↑MAT][◄].

   Result: `'.5*(2*(a^2+b^2*x^2)^1.5/3)*b^-2'`

6. Check: ['][α][←][X][ENTER][α][α][F][D][E][R][ENTER][←][SYMBOLIC][COLCT].... `'(b^2*x^2+a^2)^.5*x'` The original integrand.

Example: Use the method of substitution to find $\int x\cos(x^2)\,dx$.

1. Enter the integral with dummy limits: ['] [→] [∫] [α] [←] [A] [←] [,] [α] [←] [B] [←] [,] [α] [←] [X] [X] [COS] [α] [←] [X] [y^x] [2] [▶] [←] [,] [α] [←] [X] [ENTER].
2. Choose a transformation equation: $u = x^2$. Enter it: ['] [α] [←] [U] [←] [=] [α] [←] [X] [y^x] [2] [ENTER].
3. Perform the substitution using CHVAR: [α] [α] [C] [H] [V] [A] [R] [ENTER].

   Result: '∫(a^2,b^2,.5*COS(u),u)'
4. Use INDEF: [α] [α] [I] [N] [D] [E] [F] [ENTER] [◆]. Result: '.5*SIN(u)'
5. Substitute: [←] [{ }] [α] [←] [U] [SPC] ['] [α] [←] [X] [y^x] [2] [ENTER] [←] [SYMBOLIC] [NXT] [↑MAT] [◆]. Result: '.5*SIN(x^2)'
6. Check that: ['] [α] [←] [X] [ENTER] [α] [α] [F] [D] [E] [R] [ENTER] [←] [SYMBOLIC] [COLCT]. Result: 'COS(x^2)*x'. The original integrand.

Example: Use the method of substitution to find $\int \frac{x}{x+1}\,dx$.

1. Enter the integral, with dummy limits: ['] [→] [∫] [α] [←] [A] [←] [,] [α] [←] [B] [←] [,] [α] [←] [X] [÷] [←] [( )] [α] [←] [X] [+] [1] [▶] [←] [,] [α] [←] [X] [ENTER].
2. Choose a transformation equation: $u = x + 1$. Enter the equation: ['] [α] [←] [U] [←] [=] [α] [←] [X] [+] [1] [ENTER].
3. Perform the substitution using CHVAR: [α] [α] [C] [H] [V] [A] [R] [ENTER].

   Result: '∫(1+a, 1+b, (-1+u)/u, u)'
4. Execute INDEF: [α] [α] [I] [N] [D] [E] [F] [ENTER] [◆]. Result: '-LN(u)+u'
5. Substitute: [←] [{ }] [α] [←] [U] [SPC] ['] [α] [←] [X] [+] [1] [ENTER] [←] [SYMBOLIC] [NXT] [↑MAT] [◆]. Result: '-LN(x+1)+(x+1)'
6. Check: ['] [α] [←] [X] [ENTER] [α] [α] [F] [D] [E] [R] [ENTER] [←] [SYMBOLIC] [COLCT].... '1-INV(1+x)'

   That may not look like the original integrand, but a little algebra will show the equivalence:

$$1-\frac{1}{1+x}=\frac{(1)(1+x)-(1)(1)}{1+x}=\frac{1+x-1}{1+x}=\frac{x}{x+1}$$

Integration by Parts

The method of substitution helps to integrate derivatives produced by the chain rule. By contrast, *integration by parts* helps with derivatives produced by the product rule. This method should be considered whenever the integrand is the product of two functions where at least one of them is easy to integrate:*

$$\int f(x)g(x)\,dx$$

The integration by parts method requires that you make two substitutions: First, let the more difficult-to-integrate function be $u = f(x)$. Then you let the easy-to-integrate function be the *derivative of* v: $dv = g(x)\,dx$. After those substitutions, the integral is $\int u\,dv$, so you can use the "by-parts" formula $\int u\,dv = uv - \int v\,du$

The integration by parts method works as follows:

1. Define u and dv by making the appropriate substitutions described above.
2. Compute the derivative of u: $du = f'(x)\,dx$
3. Compute v: $v = \int g(x)\,dx$
4. Compute $uv - \int v\,du$, using the expressions for u, v, and du you've defined and computed. If the integral in this expression is no easier to compute than the original, go back to step 1 and separate the original differently.
5. Check your answer by differentiating the result and comparing it with the original integrand.

*Actually, the method of integration by parts can be used for any integrand, $f(x)$, by assuming that it's multiplied by a function, $g(x) = 1$, although this trick won't be helpful in all cases.

The program IPARTS (see page 299) automates the integration by parts method by managing the substitutions and computations. It takes the u-expression from level 3, the dv-expression (without the dx-term, which is always implied) from level 2, and the variable of integration from level 1. It returns either a fully evaluated expression containing the constant 'Ç'; a partially evaluated expression where v was computed but $\int v\,du$ was not; or a completely unevaluated expression where not even v was computable. In this last case, repeating the procedure using a different u and dv may help. In the partially evaluated case, you may be able to further the computation by using IPARTS (or other methods) with the unevaluated integral portion of the result. The examples below illustrate each of the possible results.

Example: Use integration by parts to evaluate $\int xe^x\,dx$.

1. Define u and dv. Whenever e^x is a likely option as one of the parts, it's good to define it (with dx, of course) as dv because it is its own integral. So, in this case, let $u = x$ and $dv = e^x\,dx$.
2. Enter u: ['][α][←][X][ENTER].
3. Enter dv (the dx-term is implied): ['][←][eˣ][α][←][X][ENTER].
4. Enter the variable of integration: ['][α][←][X][ENTER].
5. Compute the integral via integration by parts: [α][α][I][P][A][R][T][S][ENTER] or [VAR] ([NXT] or [←][PREV] as needed) [IPART].

   Result: 'EXP(x)*x-EXP(x)+Ç'

Example: Just out of curiosity, repeat the previous example, but swap the definitions of u and dv: This time let $u = e^x$ and $dv = x\,dx$.

1. Enter u and dv: ['][←][eˣ][α][←][X][ENTER]; ['][α][←][X][ENTER].
2. Enter the variable of integration: [ENTER].
3. Go: [α][α][I][P][A][R][T][S][ENTER] or [VAR] ([NXT] or [←][PREV]) [IPART].

   Result: '.5*EXP(x)*x^2-∫(1,x,.5*EXP(x)*x^2,x)'

   This result is more complicated than what you started with! Clearly, it is extremely important that you define u and dv properly.

Example: Use integration by parts to evaluate $\int \ln(-x)\,dx$.

1. Define u and dv. In this integrand there appears to be only one factor. Of course, you can always assume a 1 as the second factor. If you use the "one-trick," always assign it to dv. Thus, let $u = \ln(-x)$ and $dv = 1\,dx$.
2. Enter u: ['] [→] [LN] [+/–] [α] [←] [X] [ENTER].
3. Enter dv (without the dx-term, which is implied): [1] [ENTER].
4. Enter the variable of integration: ['] [α] [←] [X] [ENTER].
5. Compute the integral via integration by parts: [α] [α] [I] [P] [A] [R] [T] [S] [ENTER] or [VAR] (then [NXT] or [←] [PREV] as needed) IPART.

   Result: 'LN(-x)*x-x+C'

Example: Use integration by parts to evaluate $\int x^5 e^{-x^3}\,dx$

1. Define u and dv. In this integrand the natural break appears to be between x^5 and e^{-x^3}, but e^{-x^3} isn't an easy-to-integrate function like e^x (at least, for the HP 48). So let $u = e^{-x^3}$ and $dv = x^5\,dx$.
2. Enter u: ['] [→] [e^x] [+/–] [α] [←] [X] [y^x] [3] [ENTER].
3. Enter dv (the dx-term is implied): ['] [α] [←] [X] [y^x] [5] [ENTER].
4. Enter the variable of integration: ['] [α] [←] [X] [ENTER].
5. Compute the integral using integration by parts: [α] [α] [I] [P] [A] [R] [T] [S] [ENTER] or [VAR] ([NXT] or [←] [PREV]) IPART.

   Result: '.166666666667*EXP(-x^3)*x^6-
   ∫(1,x,-(.500000000001*EXP(-x^3)*x^8),x)'

The method failed to compute the integral; there is still an unevaluated integral expression in the result. If only it were e^x instead of e^{-x^3} ...

Example: Repeat the previous example, transforming the integral using $w=-x^3$ before applying the integration-by-parts technique.

1. Enter the integral expression, using dummy limits: ['][→][∫][α][←][A][←][,][α][←][B][←][,][α][←][X][y^x][5][×][←][e^x][+/–][α][←][X][y^x][3][▶][←][,][α][←][X][ENTER].
2. Enter the transformation equation: ['][α][←][W][←][=][+/–][α][←][X][y^x][3][ENTER].
3. Transform the integral, using CHVAR: [α][α][C][H][V][A][R][ENTER].

   Result (to 2 places): '∫(-a^3,-b^3,-(0.33*(-w)^1.00*EXP(w)),w)'

4. Now define u and dv. Let $u = w/3$ (a simplification of $-\left(\frac{1}{3}(-w)^1\right)$) and $dv = e^w dw$.
5. Enter u and dv: ['][α][←][W][÷][3][ENTER]['][←][e^x][α][←][W][ENTER].
6. Enter the variable of integration: ['][α][←][W][ENTER].
7. Apply integration by parts: [α][α][I][P][A][R][T][S][ENTER].

   Result: '0.33*EXP(w)*w-0.33*EXP(w)+C'

8. Substitute for the transformation variable: [←][{ }][α][←][W][SPC]['][+/–][α][←][X][y^x][3][ENTER][←][SYMBOLIC][NXT] ↓MAT [◀].

   Result: '0.33*EXP(-x^3)*-x^3-0.33*EXP(-x^3)+C'

9. Check the result by differentiation: ['][α][←][X][ENTER][α][α][F][D][E][R][ENTER][←][SYMBOLIC] COLCT.

   Result: '1.00*EXP(-x^3)*x^5'. The original integrand!

The previous example illustrates how much more useful the separate techniques of integration can be when used in combination with one another. But everything still depends on you, though, to sniff out the best options.

If the HP 48 can't find an exact antiderivative for an integrand, it resorts to numerical approximation, but then the exact symbolic result may get "lost" due to the machine's limited precision. Integration by parts helps find an "exact" answer.

Example: Compute the value of $\int_0^1 x^2 e^x \, dx$, first using the built-in numerical integration routine, then using IPARTS.

1. Enter it in STD display mode: [α][α][S][T][D][ENTER]['][→][∫][0][←][,] [1][←][,][α][←][X][y^x][2][×][←][e^x][α][←][X][▶][←][,][α][←][X][ENTER].
2. Estimate its numeric value: [←][→NUM]. Result: .718281828459
3. Define and enter u and dv. Let $u = x^2$ and $dv = e^x \, dx$. Note that, because you want the exact answer, use 'e^x', not 'EXP(x)': ['][α][←][X] [y^x][2][ENTER]['][α][←][E][y^x][α][←][X][ENTER].
4. Enter the variable of integration and execute IPARTS: ['][α][←][X] [ENTER][α][α][I][P][A][R][T][S][ENTER].

   Result: 'INV(LN(e))*x^2*e^x
   -∫(1,x,2/LN(e)*x*e^x,x)'
5. Once you realize that the term ln(e) = 1, the integral expression remaining in the previous result looks to be a good candidate for another integration by parts. Separate the evaluated from the unevaluated in the previous result: [PRG] LIST OBJ [◀][◀][SWAP].
6. For the unevaluated part, define and enter u and dv. Let $u = 2x$ and $dv = e^x \, dx$: ['][2][×][α][←][X] [ENTER]['][α][←][E][y^x][α][←][X][ENTER].
7. Enter the variable of integration and execute IPARTS: ['][α][←][X] [ENTER][α][α][I][P][A][R][T][S][ENTER].

   Result: '2/LN(e)*x*e^x-2/LN(e)*(e^x/LN(e))+C'
8. Subtract this result from the evaluated portion of the previous result, replace ln(e) with 1, make an extra copy of the result, and find the integral by substituting the limits for x: [−][←][{ }]['][→][LN][α][←][E] [▶][▶][1][ENTER][←][SYMBOLIC][NXT] SHOW [◀][ENTER][1]['][α][←][X] [STO][EVAL][SWAP][0]['][α][←][X][STO][EVAL][−][←][SYMBOLIC] COLCT.

   Result: '-2+e' —the exact result!

Partial Fractions

A third method for resolving symbolic integrals pertains to any integrand that is, or can be re-expressed as, a *rational fraction*—the quotient of two polynomials. The technique of *partial fractions* converts the rational fraction integrand into a polynomial plus a series of terms such as $\frac{A}{(x+r)^k}$ or $\frac{Bx+C}{\left(x^2+sx+t\right)^k}$, where the quantity $x^2 + sx + t$ cannot be factored into linear terms with real coefficients. Here are some examples to illustrate the results of *partial fraction expansion*:

$$\frac{1}{x^2-4} = \frac{1}{4x-8} - \frac{1}{4x+8}$$

$$\frac{3x^4 + x^3 + 20x^2 + 3x + 31}{(x+1)\left(x^2+4\right)^2} = \frac{2}{x+1} + \frac{x}{x^2+4} - \frac{1}{\left(x^2+4\right)^2}$$

$$\frac{x^5+2}{x^2-1} = x^3 + x - \frac{1}{2x+2} + \frac{3}{2x-2}$$

The practical details of using the method of partial fractions to compute the integral of a rational fraction, $f(x)/g(x)$, can be divided into four stages:

1. *Make sure that the degree of $f(x)$ is less than the degree of $g(x)$.* If not, then divide $g(x)$ into $f(x)$ first, keep track of the quotient, but use the *remainder of this division* as your starting rational fraction for purposes of expansion.
2. *Factor the denominator ($g(x)$) into real, irreducible polynomials of degree 2 or smaller.* Any polynomial with real coefficients can be expressed as a product of real linear and quadratic factors (though it would be difficult to perform the factorization in practice).
3. *Compute the partial fraction expansion.* This includes using the factorization of step 2 to determine the denominators of the terms of the expansion and the calculation of the coefficients in the numerators.
4. *Find the integral of the partial fraction expansion.* Each term of the expansion series will fit one of these four types with their symbolic integration patterns:

$$\int \frac{A}{x-r}\,dx \quad = A\ln|x-r| + C$$

$$\int \frac{A}{(x-r)^k}\,dx = \frac{A(x-r)^{1-k}}{1-k} + C \ \text{ for } k \neq 0,1$$

$$\int \frac{Ax+B}{x^2+sx+t}\,dx = \frac{A}{2}\left(\ln\left|x^2+sx+t\right|\right) + \frac{2B-As}{\sqrt{4t-s^2}}\tan^{-1}\left(\frac{2x+s}{\sqrt{4t-s^2}}\right) + C$$

$$\int \frac{Ax+B}{\left(x^2+sx+t\right)^k}\,dx = \frac{A\left(x^2+sx+t\right)^{1-k}}{2-2k}$$

$$+2^{2k-2}(2B-As)\left(4t-s^2\right)^{0.5-k}\left[\frac{\cos^{2k-3}\theta\sin\theta}{2k-2} + \frac{2k-3}{2k-2}\int \cos^{2k-4}\theta\,d\theta\right]$$

$$\text{where } k \neq 0,1 \text{ and } \theta = \tan^{-1}\left(\frac{2x+s}{\sqrt{4t-s^2}}\right)$$

Note that the last of these four types has a multiple quadratic in the denominator, which uses a *reduction formula* in its symbolic result, so you must repeatedly evaluate the remaining integral term until it is $\int \cos^0\theta\,d\theta$, which is $\theta + C$.

Using the HP48 for the method of partial fractions requires a set of programs. The program PFRAC (page 311) takes the symbolic rational fraction from level 1 and returns its symbolic partial fraction expansion to level 1. PFRAC converts the symbolic numerator and denominators to single polynomials, which it then factors via PFACT. Next, it computes the numerator coefficients and finally returns a list of the symbolic expanded terms.

PFRAC ends its work at the end of stage 3, allowing you then to choose your favorite approach to integration to complete the project. PFRAC relies on a number of other polynomial programs (see page 76 for a brief description of these). In addition, PFRAC relies on the program S→RF to convert a symbolic rational fraction to a polynomial numerator and denominator. You can use any of these component programs independently, of course (see the program listings in the appendix for syntax details).

Example: Find $\int \frac{x^5+2}{x^2-1}dx$, using the method of partial fractions.

1. Enter the integrand, a rational fraction, symbolically: [⬅][EQUATION] [▲][α][⬅][X][y^x][5][▶][+][2][▼][α][⬅][X][y^x][2][▶][−][1][ENTER].
2. Compute the partial fraction expansion using PFRAC: [α][α][P][F] [R][A][C][ENTER] or [VAR] (then [NXT] or [⬅][PREV] as needed) [PFRAC].

   Result: `{ 'x^3+x' '3/(2*(x-1))' '-(1/(2*(x+1)))' }`
3. Because each of the factors in the list is one of first two kinds, they should be able to be matched by the built-in pattern-matching integration routine. Combine the three factors into a single symbolic expression—the expanded integrand: [EVAL][+][+].
4. Enter the variable of integration, make a copy and purge it, and compute the integral using INDEF: ['][α][⬅][X][ENTER][ENTER][⬅][PURG] [α][α][I][N][D][E][F][ENTER][◀][⬅][SYMBOLIC][COLCT].

   Result: `'1.5*(LN(-1+x)-.5*LN(1+x)+.5*x^2+.25*x^4'`

Example: Find $\int \frac{2x^2+3}{x(x-1)^2}dx$, using the method of partial fractions.

1. Enter the integrand in its symbolic form: [⬅][EQUATION][▲][2][α][⬅] [X][y^x][2][▶][+][3][▼][α][⬅][X][X][⬅][()][α][⬅][X][−][1][▶][y^x][2] [ENTER].
2. Compute the partial fraction expansion, using PFRAC: [α][α][P][F][R] [A][C][ENTER] or [VAR] (then [NXT] or [⬅][PREV] as needed) [PFRAC].

   Result: `{ '5/(x-1)^2' '-(1/(x-1))' '3/x' }`
3. Combine the three factors into the expanded integrand: [EVAL][+][+].
4. Enter the variable of integration and find the integral using INDEF: ['][α][⬅][X][ENTER][α][α][I][N][D][E][F][ENTER][◀].

   Result: `'3*LN(x)-LN(x-1)+5*-INV(x-1)'`

Example: Find $\int \frac{3x^4 + x^3 + 20x^2 + 3x + 31}{(x+1)(x^2+4)^2} dx$, using partial fractions.

1. Enter the integrand in its symbolic form: [⬅] [EQUATION] [▲] [3] [α] [⬅] [X] [yˣ] [4] [▶] [+] [α] [⬅] [X] [yˣ] [3] [▶] [+] [2] [0] [α] [⬅] [X] [yˣ] [2] [▶] [+] [3] [α] [⬅] [X] [+] [3] [1] [▼] [⬅] [()] [α] [⬅] [X] [+] [1] [▶] [⬅] [()] [α] [⬅] [X] [yˣ] [2] [▶] [+] [4] [▶] [yˣ] [2] [ENTER].
2. Compute the partial fraction expansion using `PFRAC`: [α] [α] [P] [F] [R] [A] [C] [ENTER].

   Result: `{ '2/(x+1)' '-(1/(x^2+4)^2)' 'x/(x^2+4)' }`
3. Though there are some quadratic terms (types 3 and 4) that the built-in routine may not be able to match, proceed as before—combining the factors into the expanded integrand: [EVAL] [+] [+].
4. Enter the variable of integration and find the integral via `INDEF`: ['] [α] [⬅] [X] [ENTER] [α] [α] [I] [N] [D] [E] [F] [ENTER] [◀].

   Result: `'2*LN(x+1)+∫(1, x, x/(4+x^2)-(4+x^2)^-2, x)'`

   `INDEF` returns a partially evaluated solution. To complete the solution, you must either use the formulas given on page 191, use some combination of other integration techniques (substitution or integration by parts) or include the relevant patterns in `IPATS` via `ADDPAT` (see page 175 for details), using `XINDEF` instead of `INDEF` for the ensuing integration.
5. Using the appropriate formula for the first of the two problematic terms (type 3), let $A = 1$, $B = 0$, $s = 0$, and $t = 4$:

$$\int \frac{x}{x^2+4} dx = \frac{1}{2}\ln\left|x^2+4\right| + C$$

6. Using the appropriate formula for the second of the two problematic terms (type 4), let $A = 0$, $B = 1$, $s = 0$, $t = 4$, and $k = 2$.

$$\int \frac{1}{\left(x^2+4\right)^2} dx = \frac{1}{8}\cos\theta\sin\theta + \frac{1}{16}\theta + C = \frac{x}{8\left(x^2+4\right)} + \frac{1}{16}\left(\tan^{-1}\tfrac{x}{2}\right) + C$$

7. All together: $2\ln|x+1| + \frac{1}{2}\ln\left|x^2+4\right| - \frac{x}{8\left(x^2+4\right)} - \frac{1}{16}\tan^{-1}\frac{x}{2} + C$

5. Applications of the Integral

Measuring with Integrals

In many situations, a quantity can be expressed as a constant multiplied by the length of an interval. A rectangular area is its constant height multiplied by its length; the work applied to an object is a constant force times the distance the object moved; the distance an object travels at constant velocity is that constant velocity times the length of time it travelled. But in any of these cases, if the "constant" is not constant, but instead *varies as a continuous function* over the interval, then the definite integral is the appropriate computational tool. It allows you to "multiply" a functional variable by a fixed interval.

Thus, definite integrals provide a means of measuring the *net effect* of functional quantity over a given interval. Sometimes, this net effect is an *area* (the base definition of the definite integral); sometimes it is a volume, or length of a curve, or an average value of a function within a given interval, a probability, or some other physical measurement such as work, distance, or fluid pressure. Each of the sections in this chapter covers one of these applications.

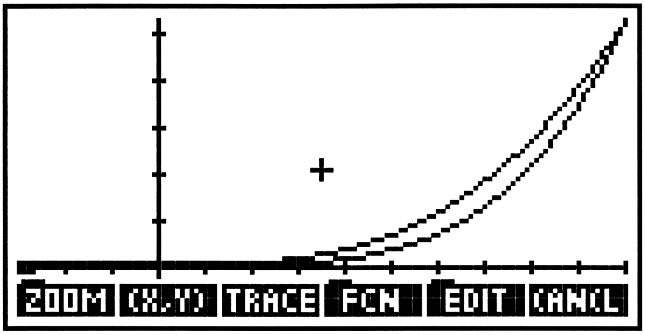
Area Between Two Curves

The integral is used to find the area of the region delineated by an interval and bounded by two functions. Often one of the functions is $f(x) = 0$—the x-axis—and the integral becomes the area of the region "under the curve" as discussed earlier. In general, to find the area between two non-zero functions that don't intersect within the interval, you can simply subtract the integral of the "smaller" function from that of the "larger." Of course, you should plot the two functions first so that you know how they interact.

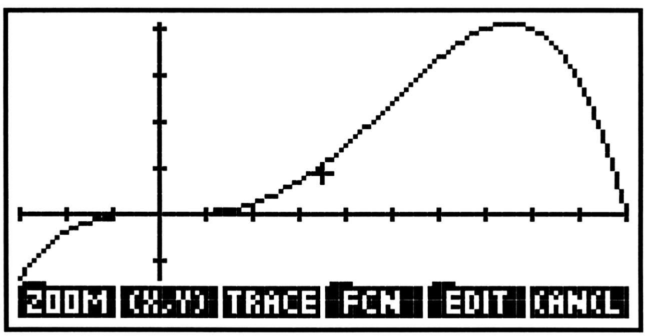
Example: Find the area bounded by the graphs of $y = x^3$ and $y = x^4$ between the limits of 0 and 1.

1. Open the PLOT application, set the TYPE: to Function, and reset the plot parameters: [→][PLOT][▲][α][F][DEL][▼][ENTER].
2. Enter a list containing the two functions in the EQ: field: [▼][←][{ }] ['][α][←][X][y^x][3][▶]['][α][←][X][y^x][4][ENTER].

3. Set INDEP: to x (lower-case), H-VIEW: to -.3 1, and V-VIEW to -.2 1: [α][←][X][ENTER][•][3][+/−][ENTER][1][ENTER][▶][•][2][+/−][ENTER][1][ENTER].
4. Draw the plot: ERASE DRAW.



5. With the cursor near the widest portion of the crescent, press TRACE [←][VIEW] to see which curve is which. Result: The upper curve is x^3.
6. Return to the PLOT screen and enter $x^3 - x^4$ into EQ:: [CANCEL][▼]['][α][←][X][y^x][3][−][α][←][X][y^x][4][ENTER].
7. Check AUTOSCALE, but leave the other plot parameters as they are and redraw the plot: [▼][◀][+/−] ERASE DRAW.



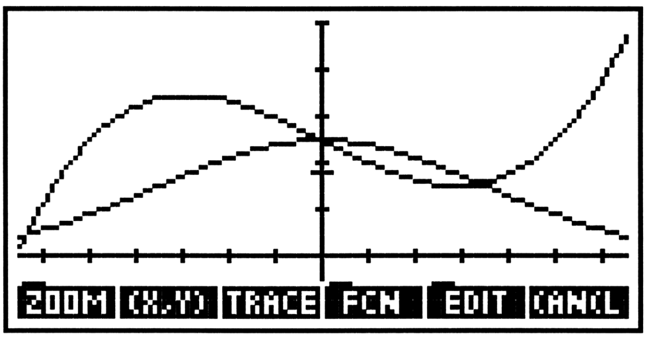
8. Compute the area under this curve (it's the same area as between the two original curve). Move the cursor to $x = 0$ (the y-axis) and press FCN AREA to mark the lower limit of the computation. Move the cursor to $x = 1$ (the right-most column of the display), then press AREA to mark the upper limit and initiate the integration.

Result: AREA: .05

Often, before computing the area of a region bounded by two functions, f and g, you may need to compute the intersection points, which represent the interval limits, a and b, of the integration. Also, if the functions intersect *within the integration interval*, the area can be computed by the integral $\int_a^b |f(x) - g(x)|\,dx$.

Example: Find the area bounded by the graphs of $y = e^{-x^2}$ and $y = x^3 - x + 1$.

1. Return to the **PLOT** application and enter a list containing the two functions in the **EQ:** field: [CANCEL] [▲] [←] [{ }] ['] [←] [e^x] [+/–] [α] [←] [X] [y^x] [2] [▶] [▶] ['] [α] [←] [X] [y^x] [3] [–] [α] [←] [X] [+] [1] [ENTER].
2. Set **H-VIEW: -1.3 1.3** and **V-VIEW -.5 2**: [α] [←] [X] [ENTER] [1] [•] [3] [+/–] [ENTER] [1] [•] [3] [ENTER] [▶] [•] [5] [+/–] [ENTER] [2] [ENTER].
3. Draw the plot: **ERASE** **DRAW**.



4. Note that the bounded area falls into two regions delineated by the three points of intersection of the two curves. Determine the three points of intersection. Move the cursor near the left-hand intersection point and press **FCN** **ISECT**. Then move the cursor near the middle intersection point and press [NXT] **ISECT**. Finally move the cursor near the right-hand intersection point and press [NXT] **ISECT**.
5. After each ISECT, the computed point is displayed (as a complex number) in the lower left portion of the plot. Press [CANCEL] [CANCEL] to return to the stack to see the three intersection points.

<u>Result</u> (to 6 places):

```
3: I-sect: (-1.276638,.195968)
2:                I-sect: (0,1)
1:    I-sect: (.676228,.633000)
```

6. Create the integral expression for the area computation using symbolic limits: ['] [α] [←] [A] [ENTER] ['] [α] [←] [B] [ENTER] [α] [α] [E] [Q] [ENTER] [EVAL] [−] [MTH] VECTR ABS ['] [α] [←] [X] [ENTER] [→] [∫].

   Result: `'∫(a,b,ABS(EXP(-x^2)-(x^3-x+1)),x)'`

7. Store the x-value of the left-most intersection point in `'a'` and that of the right-most intersection point in `'b'`: [SWAP] [MTH] [NXT] CMPL C→R [◆] ['] [α] [←] [B] [STO] [←] [STACK] ROT [→] [MENU] C→R [◆] ['] [α] [←] [A] [STO].

8. Enter the x-value of the middle intersection point as the segmentation point and use `NSEGINT` to compute the area: [0] [ENTER] [α] [α] [N] [S] [E] [G] [I] [N] [T] [ENTER]. Result: `.690192087275`

If the curve(s) you're trying to integrate are vertically oriented, such as $x = f(y)$, then you must integrate with respect to the y-axis, using a vertical interval.

Example: Find the area bounded by the graphs of $x = y^3$ and $x = 2y^3 + y^2 - 2y$.

1. Compute the intersection points:
$$y^3 = 2y^3 + y^2 - 2y$$
$$0 = y^3 + y^2 - 2y = y(y-1)(y+2)$$
$$y = \{-2, 0, 1\}$$

2. Enter the integral expression (with respect to y) using the smallest and largest intersection point y-values as the lower and upper limits respectively: ['] [→] [∫] [+/−] [2] [←] [,] [1] [←] [,] [MTH] VECTR ABS [α] [←] [Y] [y^x] [3] [−] [←] [( )] [2] [×] [α] [←] [Y] [y^x] [3] [+] [α] [←] [Y] [y^x] [2] [−] [2] [×] [α] [←] [Y] [▶] [▶] [←] [,] [α] [←] [Y] [ENTER].

3. Using y-value of the middle intersection point as the segmentation point, compute the area via `NSEGINT`: [0] [ENTER] [α] [α] [N] [S] [E] [G] [I] [N] [T] [ENTER]. Result: `3.08333333334`

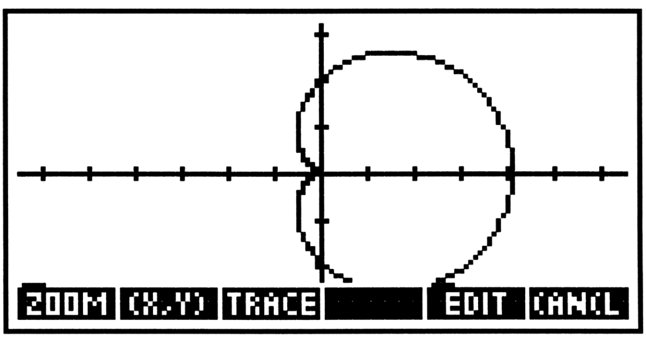
   If you rationalize this result (using `8 FIX →Q` or the program `→Qc`), you'll see that the result is `'37/12'`.

Area Bounded by Polar Functions

Functions presented in polar coordinates are integrated just as rectangular functions, but instead of using rectangles as approximations, polar integration uses circular sectors (i.e. "pie slices"). A given polar function, $r(\theta)$, is integrated by computing $\frac{1}{2}\int_a^b r^2(\theta)\,d\theta$. However, integrating polar functions presents a challenge not found with rectangular functions. Polar functions are *periodic* and thus "sweep" out the same area repeatedly. So you must keep a sharp eye on your limits of integration to be sure that you're computing only *one* period and no more.

Example: Find the total area enclosed by $r = \pi(\cos\theta - 1)$.

1. Plot the function to determine its periodicity. Open PLOT, set TYPE: to Polar and reset the parameters: [→][PLOT][▲][+/−][DEL][▼][ENTER].
2. The function in EQ: [▼]['][←][π][×][←][( )][COS][α][→][F][▶][−][1][ENTER].
3. Set INDEP: to θ, H-VIEW to -10 10, and V-VIEW to -5 5. In PLOT OPTIONS, set LO: to 0 and HI: to 6.28 (2π), then plot: [α][→][F][ENTER][1][0][+/−][ENTER][1][0][ENTER][▶][5][+/−][ENTER][5][ENTER] OPTS [▶][0][ENTER][6][•][2][8][ENTER][ENTER] ERASE DRAW.

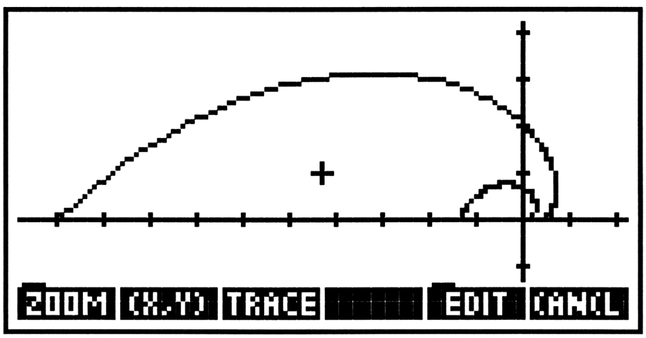


4. At the stack, enter the appropriate integral: [CANCEL][CANCEL]['][•][5][×][→][∫][0][←][,][2][×][←][π][←][,][←][( )][←][π][×][←][( )][COS][α][→][F][▶][−][1][▶][▶][y^x][2][←][,][α][→][F][ENTER].
5. Evaluate: [←][→NUM]. <u>Result</u>: 46.5094150204
   This is numerically approximate to the analytical answer, $\frac{3\pi^3}{2}$.

Just as with rectangular curves, you can find the area bounded by two polar curves (r_1 and r_2) and two values of θ, a and b, by this formula: $\frac{1}{2}\int_a^b \left| r_1^2(\theta) - r_2^2(\theta) \right| d\theta$

Example: Find the area of the region bounded by $r = e^\theta$ and $r = \theta$ between $\theta = 0$ and $\theta = \pi$. Plot the two curves first to visualize the problem.

1. Open the **PLOT** application and reset the plot parameters: [→][PLOT] [DEL][▼][ENTER].
2. Enter the two curves as a list in the **EQ:** field: [←][{ }]['][←][e^x][α][→] [F][▶][▶]['][α][→][F][ENTER].
3. Set **INDEP:** to **θ**, **H-VIEW** to **-25 5**, **V-VIEW** to **-5 10**, and in the **PLOT OPTIONS** screen, **LO: 0** and **HI: 3.15** (≈π). Then return to the main **PLOT** screen and draw the plot: [α][→][F][ENTER][2] [5][+/−][ENTER][5][ENTER][▶][5][+/−][ENTER][1][0][ENTER] **OPTS** [▶][0] [ENTER][3][·][1][5][ENTER][ENTER] **ERASE** **DRAW**.



4. Return to the stack ([CANCEL][CANCEL]) to compute the area of the "beaked" region between the curves. Note that the curve $r = e^\theta$ is always greater than $r = \theta$, so you needn't use the absolute value when entering the integral: ['][·][5][×][→][∫][0][←][,][←][π][←][,][←][e^x] [2][×][α][→][F][▶][−][α][→][F][y^x][2][←][,][α][→][F][ENTER].
5. Evaluate: [EVAL][EVAL]....(The HP 48 matches this integral!)

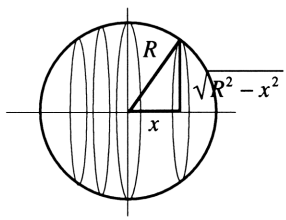
   Result: `'.5*(-(π^3/3)+EXP(2*π)/2-.5)'`

Volume

Just as the area under a curve is approximated by summing a sequence of vertical "strips" of very small width, so the volume of a region in space is approximated by summing a sequence of cross-sectional slices of very small depth. In each case, the integral is the limit of the summation as the width of the strip or the depth of the slice approaches zero. Thus, if $A(t)$ expresses the area of a cross-section taken from a region at $x = t$, then the integral to compute the volume of the region between $x = a$ and $x = b$ is $\int_a^b A(t)\,dt$.

Example: Find the volume of a sphere of radius R by integrating its cross-sectional area.

1. Determine the function describing the cross-sectional area of a sphere. The cross-section of a sphere is a circle. The radius of the circle depends on how far (x) the cross-section is from the center:



The area of the cross-sectional circle is thus $A(x) = \pi(R^2 - x^2)$.

2. Enter the integral expression using $A(x)$ as the integrand and $-R$ and R as the limits (i.e. the cross-sections run from $x = -R$ to $x = R$): ['] [→][∫] [+/−] [α][R] [←][,] [α][R] [←][,] [←][π] [×] [←][( )] [α][R] [yˣ] [2] [−] [α][←][X] [yˣ] [2] [▶] [←][,] [α][←][X] [ENTER].

3. Evaluate and simplify the results using EXCO (a program from HP's *Advanced User's Reference*—see page 294 in the Appendix here for a listing): [EVAL][EVAL] [α][α] [E][X][C][O] [ENTER].

<u>Result</u>: `'1.33333333333*R^3*π'`

This is the familiar formula for the volume of the sphere, $V = \frac{4}{3}\pi r^3$.

Solids of Revolution

Many solid objects can be modelled well by imagining that they are formed when a 2-D area is revolved around an axis, thereby sweeping out a 3-D region. For example, revolving a rectangle around the line shared by one of its sides yields a cylinder. Objects formed in this manner are called *solids of revolution.*

When finding the volumes for solids of revolution, you must pay attention to three things: The first of these requirements is an obvious one, but the others are equally vital.

1. The description of the planar region being revolved.

2. The orientation of the axis of revolution—horizontal or vertical. The nature of the integration will be different depending upon the orientation of the axis with respect to the direction of the integration interval:

   If the revolution is around a horizontal axis, each cross-section in the integration is perpendicular to the axis of revolution.

   If the revolution is around a vertical axis, each cross-section is parallel to the axis of revolution.

3. The location of the axis of revolution in relation to the planar region being revolved:

   Wherever the axis of rotation exactly coincides with the boundary of the planar region, the cross-section of the revolved object will be a disk—a filled circle.

   Wherever the axis of rotation lies at a distance from the planar region then there will be a hollow space within the cross-section of the solid of revolution that must be deducted when computing the volume. This cross-section will be a "washer."

   Wherever the axis of rotation lies within the planar region then there will be some overlapping within the cross-section that must be ignored when computing the volume. This cross-section will be a disk, but one with a smaller radius than if the axis were to intersect the region at its boundary.

Slices Perpendicular to the Axis of Revolution

Example: Find the volume of the solid obtained by revolving around the x-axis the region "under" $y = x^3$, between $x = 1$ and $x = 2$.

1. Find $A(x)$. The area being rotated here shares a side with the axis of rotation; the cross-section is a disk of "maximum" radius; in this case, $r(x) = x^3$. So its area is $A(x) = \pi\left(r(x)\right)^2 = \pi\left(x^3\right)^2 = \pi x^6$.
2. Enter the appropriate integral expression: ['] [↱] [∫] [1] [↰] [,] [2] [↰] [,] [↰] [π] [×] [α] [↰] [X] [y^x] [6] [↰] [,] [α] [↰] [X] [ENTER].
3. Set the display to `6 FIX`, evaluate, and rationalize: [6] [α] [↰] [F] [I] [X] [ENTER] [↰] [→NUM] [↰] [SYMBOLIC] [NXT] [→Qπ]. <u>Result</u>: `'127/7*π'`

Example: Find the volume of the solid obtained by revolving, around the line $y = -1$, the region "under" $y = x^3$, between $x = 1$ and $x = 2$.

1. Determine $A(x)$. The axis of revolution is outside the area being revolved; the cross-section is a washer of outer radius $R(x) = x^3 + 1$, and inner radius is $r(x) = 1$. Thus the area of the washer is:

$$A(x) = \pi\left[R(x)^2 - r(x)^2\right] = \pi\left[\left(x^3 + 1\right)^2 - (1)^2\right] = \pi\left(x^6 + 2x^3\right)$$

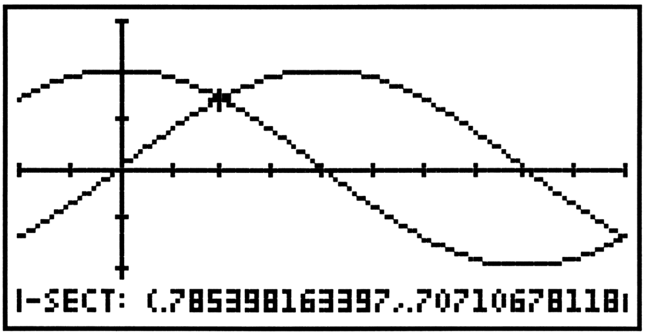
2. Enter the expression: ['] [↱] [∫] [1] [↰] [,] [2] [↰] [,] [↰] [π] [×] [↰] [( )] [α] [↰] [X] [y^x] [6] [+] [2] [×] [α] [↰] [X] [y^x] [3] [▶] [↰] [,] [α] [↰] [X] [ENTER].
3. Evaluate and rationalize: [↰] [→NUM] [↰] [SYMBOLIC] [NXT] [→Qπ].

<u>Result</u>: `'359/14*π'`

If the axis of revolution intersects both the interior and exterior of the revolved area, you need to break the problem into a series of "pure" segments—where, for each segment, the relationship between the axis of revolution and the revolved area is the same. The program `SREVH` (see page 327) manages these segment complications so that finding the integral of a solid of revolution (around a axis of revolution perpendicular to the cross-sections) is straightforward. `SREVH` uses a list on level 5, containing (in order): the lower limit, the intersection points (least to greatest), if any, in the interval, the upper limit; on levels 4 and 3, it uses the functions describing the boundaries of the revolved area (either order); level 2, the integration variable; and a real number k, where $y = k$ is the axis of revolution.*

Example: Find the volume obtained by revolving around the x-axis the region between $y = \sin x$ and $y = \cos x$, between $x = -\pi/4$ and $x = 5\pi/4$.

1. Plot the two functions and use ISECT: [→][PLOT][▲][α][F][DEL][▼][ENTER][▼][←][{ }]['][SIN][α][←][X][▶][▶]['][COS][α][←][X][ENTER][α][←][X][ENTER][•][7][8][5][4][+/−][ENTER][3][•][9][2][7][ENTER][▶][1][•][5][+/−][ENTER][1][•][5][ENTER] **OPTS** [▲][•][3][9][2][7][ENTER][•][5][ENTER][+/−][ENTER] **ERASE** **DRAW**, then **FCN** **ISECT**.



The interior point of intersection is at $x = \pi/4 \approx .785398163398$.

2. At the stack, enter the list of critical x-values: [CANCEL][CANCEL][←][{ }]['][+/−][←][π][÷][4][▶]['][←][π][÷][4][▶]['][5][÷][4][×][π][ENTER].
3. Enter the two functions representing the other boundaries of the revolved region: ['][SIN][α][←][X][ENTER]['][COS][α][←][X][ENTER].
4. In STD mode, enter the integration variable and the y-value of the axis of revolution: [α][α][S][T][D][ENTER]['][α][←][X][ENTER][0][ENTER].
5. Estimate the volume of the solid: [α][α][S][R][E][V][H][ENTER] or [VAR] ([NXT] or [←][PREV] as needed) **SREVH**. Result: 10.769685121, which matches the analytical answer, $\dfrac{3\pi^2 + 18\pi}{8}$.

*Note that, although SREVH will work for any solid of revolution whose slices are perpendicular to its axis of revolution, it's designed to be efficient for those situations where the cross-section function, $A(x)$, changes at distinct points within the integration interval. Thus, SREVH may seem to be unnecessarily slow if you use it for simpler situations where $A(x)$ is the same function throughout the integration interval.

Slices Parallel to the Axis of Revolution

If you revolve a cross-sectional slice which is parallel to the axis of revolution you get a *cylindrical shell.* Integrating a series of such parallel cross-sections approximates the volume of the solid of revolution by summing an infinitely series of thin shells, each of which has a slightly different radius.

Since the formula for the area of a cylinder is $2\pi rh$, the volume of a solid of revolution created by integrating cylindrical shells is $\int_a^b 2\pi r(x)h(x)\,dx$, where $r(x)$ describes the distance of the shell from the axis of revolution (i.e. the radius of the shell) and $h(x)$ describes the vertical height of the shell.

Example: Find the volume of the solid of revolution obtained by revolving the region "under" $y = x^3$ between $x = 1$ and $x = 2$ around the y-axis.

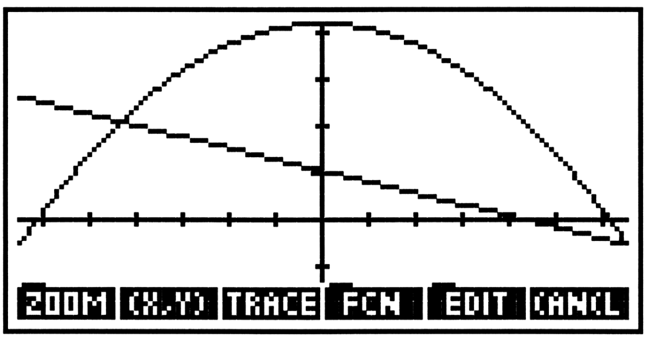
1. Analyze the problem. Each vertical slice of the revolved region traces out a cylinder whose height is x^3 and whose radius is x.
2. Enter the appropriate integral expression: ['] [→] [∫] [1] [←] [,] [2] [←] [,] [2] [×] [←] [π] [×] [α] [←] [X] [y^x] [4] [←] [,] [α] [←] [X] [ENTER].
3. The volume: [EVAL] [EVAL] [←] [SYMBOLIC] COLCT. Result: '12.4*π'

Not all rotations around the y-axis require the use of parallel cross-sections…

Example: Find the volume of the solid of revolution obtained by revolving the region bounded by $x = 8 - y^2$ and $x = 2 - y$ around the y-axis.

1. Analyze the problem. The revolved region is "vertically-oriented" (i.e. x is a function of y), and the axis of revolution is the y-axis. You can, of course, pretend that the revolved region is "horizontally oriented" and revolved around the horizontal (x-) axis, but the slices are *perpendicular* to the axis in either case. (Conclusion: This problem belongs in the previous section.) So, change the boundaries to $y = 8 - x^2$ and $y = 2 - x$, compute the limits of integration and use SREVH.

2. Plot the *revised* boundary functions and find the limits of integration and any other critical points: [→][PLOT][▲][α][F][DEL][▼][ENTER][▼][←] [{ }]['][8][−][α][←][X][y^x][2][▶]['][2][−][α][←][X][ENTER][α][←][X][ENTER] [3][+/−][ENTER][3][ENTER][▶][4][+/−][ENTER][8][ENTER] ERASE DRAW.



3. Find the x-values of the points of intersection of the curves. Note that they intersect only at the two limits. Press [◀] until the cursor is near the lower intersection point, then FCN ISECT. The lower limit: -2. Move the cursor (using [▶]) near the upper intersection point, then press [NXT] ISECT. Upper limit: 3.

4. Return to the stack and enter the two limits as a list: [CANCEL][CANCEL] [←][{ }][2][+/−][SPC][3][ENTER].

5. Enter the two boundary functions (you can use EQ from the plot): [α] [α][E][Q][ENTER][EVAL].

6. Enter the integration variable and the axis of revolution (now $y = 0$): ['][α][←][X][ENTER][0][ENTER].

7. The volume: [α][α][S][R][E][V][H][ENTER]. Result: 524.911346163

Arc Length

To measure the length of a "curve" composed exclusively of straight-line segments, you simply add up the lengths of the individual segments. Integration allows the extension of this approach to general curves by imagining that the length of the individual segments approaches zero (and consequently the number of approximating segments approaches infinity). Using integration to compute *arc length* requires that the length of an approximating line segment be expressed as a function of x. With the Pythagorean theorem it's easy to show that the arc length of $f(x)$ is $\int_a^b \sqrt{1+f'(x)^2}\,dx$, where $f'(x)$ is the derivative of $f(x)$ (i.e. the slope of the infinitely tiny line segment).

Example: Find the arc length of the curve $y = x^3$ between $x = -1$ and $x = 4$.

1. Enter the lower and upper limits: [1] [+/−] [ENTER] [4] [ENTER].
2. Enter the curve and find its derivative via FDER: ['] [α] [←] [X] [y^x] [3] [ENTER] ['] [α] [←] [X] [ENTER] [α] [α] [F] [D] [E] [R] [ENTER]. Result: '3*x^2'
3. With this result, complete the arc length integrand: [←] [x^2] [1] [+] [$\sqrt{x}$].
4. Enter the variable of integration, set the display to 5 FIX, and integrate: ['] [α] [←] [X] [ENTER] [5] [α] [α] [F] [I] [X] [ENTER] [→] [∫] [←] [→NUM].

Result: 66.21984

One of the most important uses of arc length is as a substitute for the straight-line interval of integration. Many real-world integrations are most easily handled by integrating a function with respect to its *arc length* rather than with respect to its distance from the origin along the x-axis (which is the more conventional meaning of the "interval of integration").

For example, the surface area of a *surface of revolution* created by revolving a curve around an axis can be best approximated using arc length. The surface area of each approximating band is $2\pi\, r(x)\, \Delta s$ where $r(x)$ is the distance of the curve from the axis of revolution and where Δs is the width of the band (also the length of the approximating segment). Thus, integrating so that Δs approaches zero, the surface area of a surface of revolution is $\int_a^b 2\pi r(x)\,ds$. However, since the arc length of each segment, *ds*, can be expressed in terms of *x*: $ds = \sqrt{1+f'(x)^2}\,dx$, the surface area integral is therefore $\int_a^b 2\pi r(x)\sqrt{1+f'(x)^2}\,dx$.

Finally, note that $r(x) = |f(x)|$ whenever the axis of revolution is the *x*-axis. In any case $r(x)$ is often a straight-forward modification of $f(x)$.

Example: Find the area of the surface formed by revolving the curve $y = \sin x$ between $x = 0$ and $x = 2\pi$ around the *x*-axis.

1. Analyze: Because the axis of revolution is the *x*-axis, $r(x) = |\sin x|$.
2. Enter the upper and lower limits: [0] [ENTER] ['] [2] [×] [←] [π] [ENTER].
3. Create and enter the integrand: ['] [SIN] [α] [←] [X] [ENTER] [ENTER] ['] [α] [←] [X] [ENTER] [α] [α] [F] [D] [E] [R] [ENTER] [←] [x^2] [1] [+] [√x] [SWAP] [MTH] [VECTR] [ABS] [×] [2] [×] [←] [π] [×].
4. Enter the variable of integration and compute the surface area: ['] [α] [←] [X] [ENTER] [→] [∫] [←] [→NUM].

Result: 28.84720

Example: Find the area of the surface formed by revolving the curve $y = \sin x$ between $x = 0$ and $x = 2\pi$ around the line $y = 0.5x$.

1. Analyze the situation. This time, $r(x) = |\sin x - 0.5|$.
2. Enter the upper and lower limits: [0] [ENTER] ['] [2] [×] [←] [π] [ENTER].
3. Enter the integrand and the integration variable, and compute the surface area: ['] [SIN] [α] [←] [X] [ENTER] [ENTER] ['] [α] [←] [X] [ENTER] [α] [α] [F] [D] [E] [R] [ENTER] [←] [x^2] [1] [+] [√x] [SWAP] [•] [5] [−] [MTH] [VECTR] [ABS] [×] [2] [×] [←] [π] [×] ['] [α] [←] [X] [ENTER] [→] [∫] [←] [→NUM].

Result: 33.34089

Averages

The term *average* is normally applied to a discrete, finite set of numbers: the arithmetic average for such a set of numbers is their sum divided by the size of the set. But how would you find the average of an infinitely large set of numbers? Integral calculus to the rescue once again! The problem is often recast as finding the *average value of a function* over a given interval. Since such an interval contains an infinite number of points, techniques of calculus are required.

The arithmetic average value of a function, $f(x)$ over an interval $x = a$ to $x = b$ is:

$$\frac{1}{b-a}\int_a^b f(x)\,dx$$

This is simply the total area under the curve dividing by the size of the interval.

Now, the Mean Value Theorem for Integrals requires that there exists at least one point x_0 within the interval whose function value $f(x_0)$ exactly matches the average value for the interval as a whole. To compute this value you only need to solve the following equation for x_0: $\frac{1}{b-a}\int_a^b f(x)\,dx = f(x_0)$

Example: Find the average value of sin $2x + 0.5\cos x$ for $x = 0 \le x \le \pi/4$. Then find the value of x where this average value is achieved.

1. Enter the expression for the average value: [⬅] [EQUATION] [1] [÷] [⬅] [π] [÷] [4] [▶] [▶] [➡] [∫] [0] [▶] [⬅] [π] [÷] [4] [▶] [▶] [SIN] [2] [α] [⬅] [X] [▶] [+] [•] [5] [COS] [α] [⬅] [X] [▶] [▶] [α] [⬅] [X] [ENTER].
2. Evaluate: [ENTER] [⬅] [→NUM]. Result: 1.08678 —the average value.
3. Enter the function, equate it with the average, and solve for x: ['] [SIN] [2] [X] [α] [⬅] [X] [▶] [+] [•] [5] [X] [COS] [α] [⬅] [X] [ENTER] [⬅] [=] ['] [α] [⬅] [X] [ENTER] [⬅] [{ }] [0] [SPC] ['] [⬅] [π] [÷] [4] [ENTER] [⬅] [SOLVE] [ROOT] [ROOT]. Result: 0.33045 —the x-value where $f(x)$ = average.

There are two useful ways to modify the plain arithmetic average....

The *weighted* average gives different "weights" to different portions of an interval (whereas the standard arithmetic average gives equal "weights" to all portions of an interval). The weighted average is useful whenever you're trying to find the average of some quantity that's a function of location but which varies throughout the interval. Examples of such quantities might be mass, temperature, density and so forth. If $m(x)$ describes the variation of the quantity in question, the weighted average is found by a ratio of integrals: $\dfrac{\displaystyle\int_a^b xm(x)\,dx}{\displaystyle\int_a^b m(x)\,dx}$

Example: Find the center of mass of a uniform, flat plate whose shape is the region bounded by $y = x^2$ and $y = x^3$, between $x = 0$ and $x = 2$.

1. Analyze the task. The center of mass in this context represents the x-value of the balance point for the flat plate. Imagine the flat plate positioned vertically. You are seeking the fulcrum point along its lower surface that would exactly balance the plate. In essence, you are trying to find the weighted *average* of all the vertical cross-sections for the plate. The cross-section function is $m(x) = |x^2 - x^3|$.
2. Enter the numerator integral expression and make an extra copy: ['] [→] [∫] [0] [←] [,] [2] [←] [,] [α] [←] [X] [×] [MTH] [VECTR] [ABS] [α] [←] [X] [y^x] [2] [−] [α] [←] [X] [y^x] [3] [▶] [←] [,] [α] [←] [X] [ENTER] [ENTER].
3. Because of the absolute value in the integrand, evaluate the integral after segmenting it at the point within the interval where the absolute value term equals zero (at $x = 1$ here); use `NSEGINT`: [1] [α] [α] [N] [S] [E] [G] [I] [N] [T] [ENTER]. Result: `2.5`
4. Swap the other copy of the integral expression into level 1 and edit it to remove the factor x from the integrand: [SWAP] [←] [EDIT] [▶] [▶] [▶] [▶] [▶] [▶] [▶] [DEL] [DEL] [ENTER].
5. Evaluate the denominator integral with `NSEGINT`: [1] [ENTER] [α] [α] [N] [S] [E] [G] [I] [N] [T] [ENTER]. Result: `1.5`
6. Divide to complete the computation: [÷] [←] [SYMBOLIC] [NXT] [→Q]. Result: `'5/3'` Place the fulcrum at $x = 5/3$ to balance the plate.

The *moving* average "smooths out" functions that have a lot of short-term ups and downs. It replaces the actual value of a function with the average value of the function over some fixed previous period. If A is the size of the fixed previous period and the original function is $f(x)$, then the moving average is computed by:

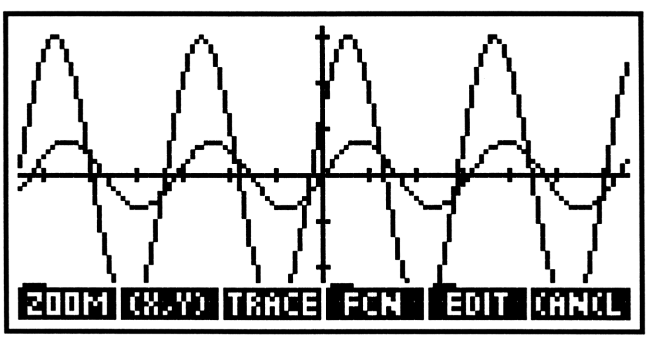
$$\frac{1}{A}\int_{x-A}^{x} f(t)\,dt$$

If you graph the original function and its moving average together, you'll see that the moving average reduces variability and exposes underlying trends.

Example: Find the moving average of $f(x) = 3\sin(2x + 0.5)$ over a period of 2 and plot it simultaneously with $f(x)$.

1. Enter the moving average integral expression: [′] [1] [÷] [2] [×] [→] [∫] [α] [←] [X] [−] [•] [5] [←] [,] [α] [←] [X] [←] [,] [3] [×] [SIN] [2] [×] [α] [←] [T] [+] [•] [5] [▶] [←] [,] [α] [←] [T] [ENTER].
2. Purge X, evaluate the integral, and tidy up the results. Note that the moving average is a symbolic integration, so it requires that the integrand be symbolically evaluable by the HP 48: [′] [α] [←] [X] [←] [PURG] [EVAL] [EVAL] [←] [SYMBOLIC] [COLCT] [EXPA] [COLCT] [EXPA] [COLCT].

   <u>Result</u>: '.75*COS(-.5+2*x)-.75*COS(.5+2*x)'
3. Enter the original integrand, combine it into a list with the moving average you just computed, enter the list into EQ, and then plot the two functions together: [′] [3] [×] [SIN] [2] [×] [α] [←] [X] [+] [•] [5] [ENTER] [2] [PRG] [LIST] [→LIST] [←] [PLOT] [←] [EQ] [→] [PLOT] [▲] [α] [F] [DEL] [▼] [ENTER] [▼] [▼] [α] [←] [X] [ENTER] [ERASE] [DRAW].



Probability

As with averages in the previous section, probability computations come in two varieties—discrete and continuous. Discrete probability is useful whenever you can compute a finite number of successful outcomes and a finite number of total outcomes—say, 200 "heads" in 256 total coin-flips. However, many probabilities are best modeled using *area*: the computation reduces to finding the number of points within a particular region and comparing it to the number of points in the total area. Since both of these quantities are infinite, you must turn to integral calculus to correctly compute such a ratio.

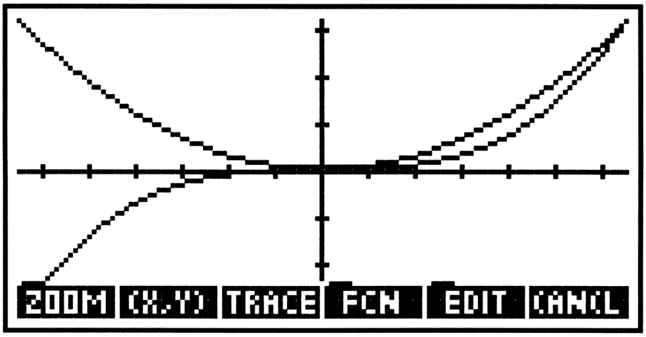
Note, however, that there is a non-calculus means of approximating continuous probabilities—modelling a continuous computation as a discrete computation. The Monte Carlo method randomly chooses a point from within the total area and determines whether or not it lies within a given region of interest. Repeating this many times offers an estimate of the probability as $\frac{\text{number of points within the region}}{\text{total number of points sampled}}$.

Using the Monte Carlo method on the HP 48 would, in fact, be a relatively easy way to solve continuous probability problems—except that it takes a *very large* number of points (and thus a large amount of time) to estimate the probability to a sufficient degree of accuracy. The Monte Carlo method is thus reserved in practice for situations where no other means are available.

Turning, therefore, to the calculus-based analytical solution, if it's area you're trying to compute, then integration is your weapon. The estimate of a continuous probability, modeled in terms of area, would clearly be $\frac{\text{area of the region}}{\text{total area}}$.

Example: A point (x,y) is randomly chosen from the square represented by $-1 \le x \le 1$ and $-1 \le y \le 1$. What is the probability that $x^3 < y < x^2$?

1. Plot the two functions within the given square: [→] [PLOT] [←] [{ }] ['] [α] [←] [X] [y^x] [2] [▶] ['] [α] [←] [X] [y^x] [3] [ENTER] [α] [←] [X] [ENTER] [1] [+/−] [ENTER] [1] [ENTER] [▶] [1] [+/−] [ENTER] [1] [ENTER] ERASE DRAW.



2. Back at the stack, find the area of the small region by computing the area between the two curves (i.e. $x^2 - x^3$) within the total square. Enter the integral expression and evaluate: [CANCEL] [CANCEL] ['] [↱] [∫] [1] [+/−] [↰] [,] [1] [↰] [,] [α] [↰] [X] [y^x] [2] [−] [α] [↰] [X] [y^x] [3] [↰] [,] [α] [↰] [X] [ENTER] [↰] [→NUM].

   Result: .666666666667 The area of the region is 2/3.

3. Compute the total area—the area of the square. The square is 2 units by 2 units or 4. Enter the area: [4] [ENTER]

4. Divide to find the probability: [÷] Result: .166666666667 The probability is 1/6.

Now suppose that, in the previous example, you want to describe the probability that, given a particular value of x, a randomly chosen value of y would lie between the two curves. Some values of x (near $x = -1$, for example) yield a high probability, while others (near $x = 0$, for example) yield a low probability.

In fact, if you create a new function, $p(x)$, that returns the probability of a successful outcome for a given value of x—scaling it properly so that its integral over all allowed values of x equals 1—this is a *probability density function.* It describes the *distribution* of probabilities with respect to the value of x.

Once you have a probability density function, computing a particular probability involves only a simple integration: probability is the area under the probability density function. The *total area* under the density function must, of course, equal 1, since the particular probability—the area under the function between two specific limits—can never exceed 1; you can never be more than 100% certain.

Many real world probabilities can be effectively modeled using a handful of very useful probability density functions. Perhaps the most common is the *normal*, or *Gaussian*, density function—the "bell curve:"

$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

where μ is the mean and σ is the standard deviation of the set of points or data being examined.

Example: What is the probability that a randomly chosen variable that's normally distributed with a mean of –2 and a standard deviation of 3 lies between 0 and 5?

1. Enter the lower and upper limits: [0] [ENTER] [5] [ENTER].
2. Enter the integrand, the normal density function ($\sigma = 3$ and $\mu = -2$): [←] [EQUATION] [1] [÷] [3] [√x] [2] [←] [π] [▶] [▶] [←] [eˣ] [+/–] [▲] [←] [( )] [α] [←] [X] [+] [2] [▶] [yˣ] [2] [▶] [▶] [1] [8] [ENTER].
3. Enter the variable of integration and evaluate the integral: ['] [α] [←] [X] [ENTER] [→] [∫] [←] [→NUM]. <u>Result</u>: .242677208919

   There's a little less than a 25% chance that a randomly chosen value with the given distribution lies between 0 and 5.

The HP 48 has four probability density functions built-in that you can access using commands (there's one command for each density function). Each of these commands computes the probability that, given certain distribution parameters, a random variable is *greater* than a given value. That is, they compute the area under the probability density function between the given value and positive infinity (an improper integral)—an area known as the *upper-tail* of the function. The area between two finite values is equal to the area between the upper-tail above the lower value minus the upper-tail above the higher value.

Thus the previous example can be solved more quickly and easily using the UTPN command (the Upper-Tail Probability—Normal distribution command). Try it.

Example: Repeat the previous example using the UTPN command.

1. Enter the mean and variance (variance is the square of the standard deviation) of the distribution: 2 +/– ENTER 9 ENTER.
2. Enter the lower limit and use UTPN: 0 ENTER MTH NXT PROB NXT UTPN. Result: .252492537547
3. Repeat steps 1 and 2 using the upper limit instead: 2 +/– ENTER 9 ENTER 5 ENTER UTPN. Result: 9.81532892865E-3
4. Subtract: –. Result: .242677208918

The other probability density functions built into the HP 48 are:*

- *Student's t-distribution (UTPT)*: $p(x)=\dfrac{\left(\frac{n+1}{2}-1\right)!}{\left(\frac{n}{2}-1\right)!\sqrt{n\pi}}\left(1+\dfrac{x^2}{n}\right)^{-\frac{n+1}{2}}$,

  where n is the degrees of freedom (a positive integer) for the distribution;

- *Chi-Square distribution (UTPC)*: $p(x)=\begin{cases} x<0 & 1 \\ x\geq 0 & \dfrac{x^{\frac{n}{2}-1}e^{-\frac{x}{2}}}{2^{\frac{n}{2}}\left(\frac{n}{2}-1\right)!} \end{cases}$,

  where n is the degrees of freedom (a positive integer) for the distribution;

- *Snedecor's F-distribution (UTPF)*:

$$p(x)=\begin{cases} x<0 & 1 \\ x\geq 0 & \left(\dfrac{n_1}{n_2}\right)^{\frac{n_1}{2}}\left(1+\dfrac{n_1 x}{n_2}\right)^{-\frac{n_1+n_2}{2}}\left(\dfrac{t^{\frac{n_1-2}{2}}\left(\frac{n_1+n_2}{2}-1\right)!}{\left(\frac{n_1}{2}-1\right)!\left(\frac{n_2}{2}-1\right)!}\right) \end{cases},$$

  where n_1 and n_2 are the degrees of freedom (a positive integer) for the numerator and denominator distributions.

*Note the fifth command, NDIST. Instead of computing the area under the function *above* a given value (like the four upper-tail commands), NDIST evaluates the function *at* the value. Thus, use UTPN to compute a probability and NDIST to compute a probability density.

Other Physical Measurements

This section discusses the application of integration to a variety of physical measurements—distance, work, and fluid pressure. The role the integral plays in each of these measurements is evident when you look at their definitions:

- Net distance$= \int_a^b v(t)\,dt$, where $v(t)$ is the object's velocity at time t.
- Total distance $= \int_a^b |v(t)|\,dt$, where $v(t)$ is the object's velocity at time t.
- Work$= \int_a^b F(x)\,dx$ where $F(x)$ is the force exerted on an object in moving it from $x = a$ to $x = b$. If this motion stretches or compresses a spring, then $F(\mathrm{x}) = kx$ (where k is the spring constant), so Work$= k\int_a^b x\,dx$. If this motion lifts against gravity, then $F(x) = w(x)x$ where $w(x)$ is the weight of the object, and x is the vertical distance between the object and its final height, so Work $= \int_a^b w(x)x\,dx$.
- Fluid Pressure $= w\int_a^b d(h)l(h)\,dh$, where w is the fluid's weight-density, $d(h)$ the fluid's depth at a distance h from the intersection of its surface and the container wall, and $l(h)$ is the width of the container wall at h.

The following examples illustrate these concepts and also serve as models for integration with unit objects on the HP 48.* With numerical integration, the units of the lower limit are used during integration, so they must be dimensionally consistent with those of both the integrand the upper limit. The units of the result are the product of the units of the integrand and the units of the lower limit.

*Symbolic integration with unit objects isn't recommended because the evaluation process for some integrands requires dimensionless values.

Example: An object moves horizontally according to the velocity function (in feet), $v(t) = -8t^2 + 32$. Find the net and total distance traveled by the object during the interval $1 \le t \le 3$.

1. Enter the integral expression for the net distance and make an extra copy: ['] [↱] [∫] [1] [↰] [,] [3] [↰] [,] [+/-] [8] [×] [α] [↰] [T] [yˣ] [2] [+] [3] [2] [↰] [,] [α] [↰] [T] [ENTER] [ENTER].

2. Compute the net distance: [EVAL] [EVAL].

   Result: `-5.33333333333`

3. Drop the result and edit the copy to include the absolute value function in the integrand: [⬅] [↰] [EDIT] [▶] [▶] [▶] [▶] [▶] [▶] [▶] [MTH] [VECTR] [ABS] [DEL] [▶] [▶] [▶] [▶] [▶] [▶] [▶] [▶] [▶] [▶] [↰] [( )] [⬅] [ENTER].

4. Compute the total distance: [↰] [→NUM].

   Result: `32`

Example: Find the work required to move a particle horizontally from $x = 3$ meters to $x = 8$ meters, if the force exerted on the particle at x is $x^3 - x + 3$ Newtons.

1. Enter the integral expression for Work, using unit objects: ['] [↱] [∫] [3] [↱] [_] [α] [↰] [M] [↰] [,] [8] [↱] [_] [α] [↰] [M] [↰] [,] [↰] [( )] [α] [↰] [X] [yˣ] [3] [−] [α] [↰] [X] [+] [3] [▶] [↱] [_] [α] [N] [↰] [,] [α] [↰] [X] [ENTER].

2. Evaluate the integral: [EVAL] [EVAL].

   Result: `- Error: Inconsistent Units`

   This is a good example of a very simple-looking integral that nevertheless fails with unit objects.

3. Drop the results of the error and repeat steps 1 and 2 without using unit objects: [⬅] [⬅] ['] [↱] [∫] [3] [↰] [,] [8] [↰] [,] [α] [↰] [X] [yˣ] [3] [−] [α] [↰] [X] [+] [3] [↰] [,] [α] [↰] [X] [ENTER] [EVAL] [EVAL].

   Result: `991.25`. With the proper units: 991.25 N•m.

Example: If one hangs a spring ($k = 10$ lbs./ft) vertically (i.e. subject to gravity) and attaches a 10 lb. weight to the spring, how much work is done in raising the weight 6 inches from where it hangs naturally?

1. Analyze the task: The total work done is that of lifting the weight against gravity, plus that of compressing the spring 6 inches from its natural position: $W_t = W_{grav} + W_{comp} = wd + k\int_0^d x\,dx$, where w is the weight being lifted, and d the distance the weight moves (in ft). Here, $w = 10$ lb, $d = 0.5$ ft, and thus W is in ft-lbs.
2. Create the expression with units: ['][↱][∫][0][↱][_][α][↰][F][α][↰][T][↰][,][•][5][↱][_][α][↰][F][α][↰][T][↰][,][α][↰][X][↰][,][α][↰][X][ENTER][1][0][_][α][↰][L][α][↰][B][÷][α][↰][F][α][↰][T][ENTER][×][1][0][↱][_][α][↰][L][α][↰][B][ENTER][•][5][↱][_][α][↰][F][α][↰][T][ENTER][×][+].
3. Evaluate the expression: [EVAL][EVAL].

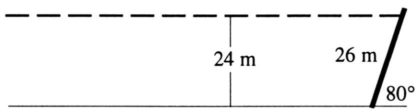
   Result: `6.25_lb*ft`. Sometimes, using unit objects works!

Example: Oil of density 50 lbs/ft^3 is 3 feet deep in a hemispherical (bowl-shaped) reservoir of radius 4 feet. You wish to pump out oil down to a depth of 1 foot. How much work will it take to pump that much oil to a point 2 feet vertically above the top of the reservoir?

1. Analyze the task. You must find $w(x)$ the weight of a layer of oil that's d feet below the top of the tank (or $x = d + 2$ below the high-point for the pumping process). Each layer has a circular surface of radius r, where $d^2 + r^2 = 4$. Therefore, each layer has a cross-sectional area of $\pi r^2 = \pi(16 - x^2)$. Finally, the weight of each layer, whose depth is dx is multiplied by the oil density, yielding:
$$w(x) = 50\pi\left(16 - d^2\right)dx = 50\pi\left(16 - (x-2)^2\right)dx$$
2. Enter the integral expression, in terms of x. Remember that the limits are measured from the point 2 feet *above* the tank: ['][↱][∫][3][↰][,][5][↰][,][5][0][×][↰][π][×][↰][( )][1][6][−][↰][( )][α][↰][X][−][2][▶][y^x][2][▶][×][α][↰][X][↰][,][α][↰][X][ENTER].
3. Evaluate: [↰][→NUM]. Result: `14241.8866963` ft-lbs.

Example: The face of a dam spanning a 100 m river is a rectangle 26 m high and inclined at an angle of 10° from vertical. Find the force (in Newtons) due to water pressure on the dam when the river is 24 feet deep. Use 9818 N/m^3 as the weight-density of water.

1. Analyze the task. Draw a brief diagram of the situation:



   If h is the distance along the dam below the surface of the river, then the depth of the river at that point is $d(h) = h \sin 80°$. The width of the dam face is constant, so $l(h) = 100$ m. The limits of integration $h = 0$ to $h = 24/\sin 80°$.

2. Enter the limits, making sure that you're in Deg mode first: [⬅][RAD] (if necessary), [0][ENTER][2][4][ENTER][8][0][SIN][÷].

3. Enter the integrand and the limits of integration: ['][α][⬅][H][ENTER] [8][0][SIN][×][1][0][0][×]['][α][⬅][H][ENTER].

4. Compute the integral: [➡][∫][EVAL]. Result: 29244.2864225

   This is the volume of water (in m^3) impinging on the dam face.

5. Multiply by the weight-density of water: [9][8][1][8][×].

   Result: 287120404.096 Newtons—over 30,000 tons.

6. Multivariate and Vector Calculus

Scalars and Vectors and Multi-Variable Functions

So far in this book, you've seen functions of only a single variable, but functions of two and three variables are also useful and common in real-world applications. This chapter explores the application of calculus—both differential and integral —to multi-variable functions of two types: *scalar-valued* and *vector-valued.*

A *scalar* is a single number; a scalar-valued function is a function whose output is a single number. Regular Cartesian functions, such as $z = x^3y - y^3$, are scalar-valued functions. By contrast, a *vector* is a *set* of numbers (or *components*); a vector-valued function's output is a vector. Vector-valued functions are usually expressed in *parametric* form, e.g, $\mathbf{r}(u,v) = [x(u,v),\ y(u,v),\ z(u,v)]$, where the components are all themselves multi-variable functions.

The techniques of calculus—differentiation and integration—apply to multivariable functions just as well as to single-variable functions, but it's much harder to visualize multi-variable functions using two dimensions. The HP48G/GX comes with some tools to help you to visualize functions of two variables (as you'll see below), but not for functions of three or more variables.*

After introducing some basic tools for working with vectors and their functions, this chapter explores the HP 48's tools for visualizing functions of two variables. The rest of the chapter examines differential and integral calculus with multi-variable functions—both scalar functions and vector functions.

*Indeed, there are no good methods for visualizing functions of three or more variables anywhere because such a task requires a four-dimensional representation! Fortunately, the vector form for functions makes it easy to work with functions of three or more variables even when it's impossible to construct visual models of them.

Vector Basics

In geometric terms, a vector is a directed line *segment*, with a finite *length* or *magnitude* (also called its *absolute value*). The *direction* of a vector is denoted by its two endpoints, the initial and the terminal endpoints, respectively. For example, the vector from point A to point B might be referred to as $\vec{AB}$ (whereas the vector from point B to point A is denoted as $\vec{BA}$).

If you assume that the initial point is always the origin (0,0,0), then vectors are especially useful to describe *points*. The point (-3,7,2), for example, can be described as a line segment—a vector—directed from the origin (0,0,0) to (-3,7,2). And since its coordinates form a set of instructions on how to reach it from the origin,* the notation [-3 7 2] offers a more algebraic (and therefore analytic) description of the vector than does the geometric description, $\vec{AB}$.

The notation used for vectors is purposely like that of matrices, because they behave algebraically like $1 \times n$ (or $n \times 1$) matrices, a trait that makes vectors powerful for both analytic geometry and multi-variable calculus. A matrix can be treated as a vector of vectors; each row or each column of a matrix is itself a vector, so the HP 48 uses [ ] for both matrices and vectors (together called *arrays)*.

Symbolic Vectors

The HP 48 *requires* that all vectors (collections within square-bracket delimiters) be purely numbers—no algebraic expressions, text, or other object types. This limits the utility of its vector data type with calculus because so many of the tasks require symbolic processing. So list braces, { }, are used instead to designate a *symbolic vector*—which contains symbolic expressions.

The creation of symbolic vectors also requires that a set of symbolic vector tools be created to perform the kinds of vector operations that the HP 48 provides for purely numeric vectors. Thus, as each vector operation is introduced, its parallel symbolic vector operation will be demonstrated.

*Any vector in three-dimensional space can be treated as the sum of three basis vectors, each running from the origin along one of the coordinate axes. The length of each basis vector is a component of the vector: The vector [-3 7 2], for example, has an x-component of -3, a y-component of 7 and a z-component of 2.

Vector Operations

The basic vector operations—addition, subtraction, and scalar multiplication—work just like their equivalent matrix operations....

Example: Add the two vectors [4 9 -1] and [3 -1 2].

1. Enter the two vectors onto the stack: ⬅ [ ] 4 SPC 9 SPC 1 +/– ENTER ⬅ [ ] 3 SPC 1 +/– SPC 2 ENTER.
2. Add: +. Result: `[ 7 8 1 ]`

Example: Add the two symbolic vectors { *a b c* } and { *x y z* }.

1. Enter the two vectors to the stack: ⬅ { } α ⬅ A SPC α ⬅ B SPC α ⬅ C ENTER ⬅ { } α ⬅ X SPC α ⬅ Y SPC α ⬅ Z ENTER.
2. Add: MTH **LIST** **ADD**. Result: `{ 'a+x' 'b+y' 'c+z' }`

   The element-wise list addition (remember that symbolic vectors are lists) requires that you use the `ADD` command—not +, which would have caused the two lists to concatenate, instead.

Example: Subtract the vector [3 -1 2] from the vector [4 9 -1].

1. Enter the vector [4 9 -1]: ⬅ [ ] 4 SPC 9 SPC 1 +/– ENTER.
2. Enter the vector [3 -1 2]: ⬅ [ ] 3 SPC 1 +/– SPC 2 ENTER.
3. Subtract: –. Result: `[ 1 10 -3 ]`

Example: Subtract the symbolic vector { *x y z* } from the vector [4 9 -1].

1. Enter the vector [4 9 -1] as a symbolic vector: ⬅ { } 4 SPC 9 SPC 1 +/– ENTER.
2. Enter the symbolic vector { *x y z*}: ⬅ { } α ⬅ X SPC α ⬅ Y SPC α ⬅ Z ENTER.
3. Subtract: –. Result: `{ '4-x' '9-y' '-1-z' }`.

   Note that numeric vectors can be combined with symbolic vectors *as long as both are expressed as symbolic vectors.*

Example: Multiply the vector [4 9 -1] by the scalar 5.3.

1. Enter the vector [4 9 -1]: ⬅[ ] 4 SPC 9 SPC 1 +/− ENTER.
2. Multiply by 5.3: 5 • 3 ×. Result: [21.2 47.7 -5.3]

Example: Multiply the vector [4 9 -1] by the symbolic scalar *a*.

1. Enter the vector [4 9 -1] in symbolic form: ⬅{ } 4 SPC 9 SPC 1 +/− ENTER.
2. Enter the symbolic scalar and multiply: ' α ⬅ A ENTER ×.
   Result: { '4*a' '9*a' '-a' }

"Multiplying" two vectors is not analogous to arithmetic. There are two kinds of vector products: The vector *dot product* is defined for any two vectors having the same number of elements. Given two vectors **r**=$[r_x\ r_y\ r_z]$ and **s**=$[s_x\ s_y\ s_z]$, the dot product, **r** • **s**, is $r_x s_x + r_y s_y + r_z s_z$. The HP 48 has a built-in command for this.

Example: Find the dot product of [4 9 -1] and [5 -3 2].

1. Enter the first vector: ⬅[ ] 4 SPC 9 SPC 1 +/− ENTER.
2. Enter the second vector: ⬅[ ] 5 SPC 3 +/− SPC 2 ENTER.
3. Compute the dot product: MTH VECTR DOT. Result: -9

Example: Find the symbolic dot product of { 4 9 -1 } and {*x y z* }.

1. Enter the first vector: ⬅{ } 4 SPC 9 SPC 1 +/− ENTER.
2. Enter the second vector: ⬅{ } α ⬅ X SPC ⬅ ⬅ Y SPC α ⬅ Z ENTER.
3. Compute the dot product using the program, SDOT (see page 322): α α S D O T ENTER. Result: '4*x+9*y-z'

(Note how similar the dot product process is to that of computing each element in a matrix multiplication: the first vector is treated as a "row," the second as a "column"—and the result is a single number.)

By contrast, the *cross product* of two vectors is another vector—perpendicular to both of the other vectors (assuming all three vectors originate at the same point): Given two vectors, **r**=[r_x r_y r_z] and **s**=[s_x s_y s_z], their cross product, **r X s**, is the vector [$r_y s_z - r_z s_y$ $r_z s_x - r_x s_z$ $r_x s_y - r_y s_x$].

Example: Find the cross product of [4 9 -1] and [5 -3 2].

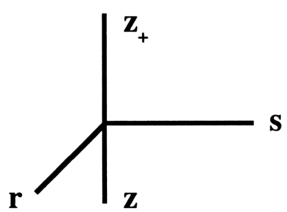
1. Enter the first vector: ⬅ [ ] 4 SPC 9 SPC 1 +/− ENTER.
2. Enter the second vector: ⬅ [ ] 5 SPC 3 +/− SPC 2 ENTER.
3. Find the cross product: CROSS. Result: `[ 15 -13 -57 ]`

Example: Find the symbolic cross product of { 4 9 -1 } and { x y z }.

1. The first vector: ⬅ { } 4 SPC 9 SPC 1 +/− ENTER.
2. The second vector: ⬅ { } α ⬅ X SPC α ⬅ Y SPC α ⬅ Z ENTER.
3. Compute the symbolic cross product, using the program `SCROSS` (see page 321): α α S C R O S S ENTER.

   Result: `{ 'y+9*z' '-x-4*z' '-(9*x)+4*y' }`

Like matrix multiplication, the cross product is not commutative. When taking the cross product **r X s**, you will get the $\mathbf{z}_+$ vector; when taking the other cross product, **s X r**, you will get the $\mathbf{z}_-$ vector:



Example: Find the cross product of [5 -3 2] with [4 9 -1].

1. Enter the first vector: ⬅ [ ] 5 SPC 3 +/− SPC 2 ENTER.
2. Enter the second vector: ⬅ [ ] 4 SPC 9 SPC 1 +/− ENTER.
3. Find the cross product: CROSS. Result: `[ -15 13 57 ]`
   This is the negative of the earlier result.

Vector Angles and Magnitudes

A vector has both magnitude (length) and direction. It should therefore be possible to find these parameters easily for a vector entered in standard form.

Example: Find the length of the vectors [4 9 -1] and { 3*x* 9 *z* }.

1. First try the numeric vector: ⬅ [ ] 4 SPC 9 SPC 1 +/– ENTER.
2. Find its length: MTH VECTR ABS. Result: 9.89949493661
3. Now do the symbolic vector: ⬅ { } ' 3 × α ⬅ X ▶ 9 SPC α ⬅ Z ENTER.
4. Find its length with the program VABS (see page 332): α α V A B S ENTER. Result: '√((3*x)^2+81+z^2)'

Finding the "direction" of a vector is more complicated. You must first decide the reference directions against which to measure the angle. In three dimensions, you use the three coordinate axes as your reference directions. The direction angles for a vector V are computed from its components (v_x, v_y, and v_z) and its length:

$$\theta_x = \cos^{-1}\frac{v_x}{|V|} \qquad \theta_y = \cos^{-1}\frac{v_y}{|V|} \qquad \theta_z = \cos^{-1}\frac{v_z}{|V|}$$

Example: Find the direction angles of the vectors [4 9 -1] and { *x* *y* *z* }.

1. In DEG mode (use ⬅ RAD, if necessary), SWAP the previous numeric result to level 1 and copy that magnitude twice: ENTER ENTER.
2. Find the *x*-, *y*-, and *z*- direction angles, respectively: 4 ENTER SWAP ÷ ⬅ ACOS (Result: 66.1677009381); ⬅ STACK ROT 9 ENTER SWAP ÷ ⬅ ACOS (Result: 24.613597653); ROT 1 +/– ENTER SWAP ÷ ⬅ ACOS. Result: 95.7976363295.
3. The other vector: ⬅ { } α ⬅ X SPC α ⬅ Y SPC α ⬅ Z ENTER.
4. Execute VDIR: α α V D I R ENTER or VAR (NXT or ⬅ PREV) VDIR. Result: { 'ACOS(x/√(x^2+y^2+z^2))'
   'ACOS(y/√(x^2+y^2+z^2))'
   'ACOS(z/√(x^2+y^2+z^2))' }

Visualizing Two-Variable Scalar Functions

In general, the graph of a two-variable function is a *surface in space*. If it is generated by a true function, $z = f(x,y)$, the surface passes the vertical line test—any vertical line only intersects the surface once. But how can you display a surface in space (a three-dimensional object) in *two* dimensions—such as on your HP 48's display or on a piece of paper? Either you must project the three dimensions onto a flat surface or hold one of the dimensions constant while plotting the other two. Neither choice is ideal because the true shape of the function will be either be distorted or partial when displayed in two dimensions. However, given these limitations, it's usually better to see something rather than nothing.

The HP 48 offers three approaches to visualizing two-variable scalar functions:

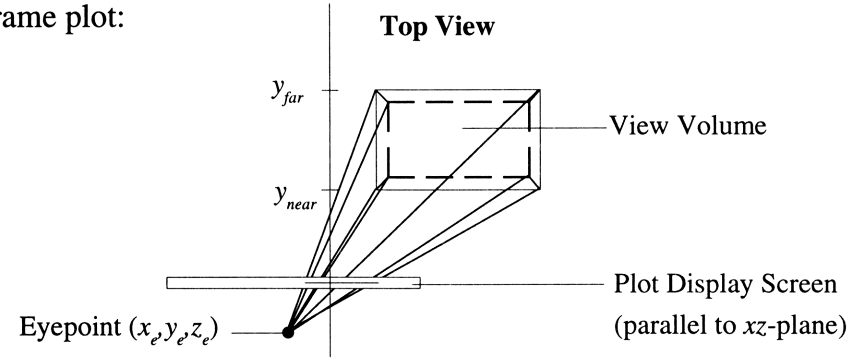
- A Wireframe plot uses a perspective projection to map a designated portion of the three-dimensional surface—expressed by a Cartesian function—onto a two-dimensional display.
- A Y-Slice plot freezes one of the independent variables and takes a two- dimensional snapshot of the resulting "slice" of the surface. Y-Slice actually takes a series of "snapshots," changing the value of the frozen variable between "shots," then playing back the series as a movie when it's finished.
- A Pseudo-Contour plot computes the derivative of the given function and then plots its slopefield—a grid of line segments whose orientation matches the slope of the function at each point in the grid. This plot is a kind of visual approximation of a contour plot where a series of level curves, $f(x,y) = k$, where the constant k is different in each curve. A contour plot (like a contour map) is a two-dimensional representation of a three-dimensional surface (such as a mountainous terrain).

All of the plots above use a grid of points—or *sampling grid*—as inputs for the function, using the coordinates of each point as the values of the two independent variables.* They differ in how they use the inputs and how they display the output.

*The two independent variables are those listed in the INDEP and DEPND fields in the plot set-up screens. The dependent variable is simply implied as the value of the function. Don't be mislead by the potentially confusing designation of one of the independent variables as DEPND.

Wireframe Plots

The Wireframe plot takes the points in the sampling grid, uses the given (Cartesian) function to compute the third coordinate, then converts all 3-D result points within the *view volume* into 2-D points via a perspective projection centered on the *eyepoint*. Finally, each point is connected to its neighbors with a line segment to display the surface as a "wire-mesh." Here is a top view of what happens during a wireframe plot:

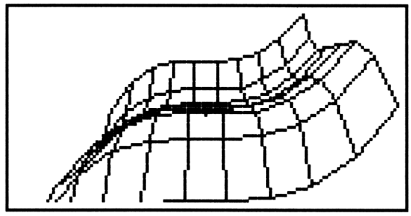


Note several important points about this diagram and the relationships it depicts:

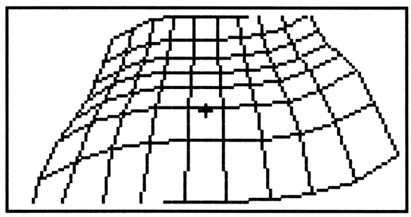
- The plot display screen and eyepoint are "frozen" together; when you move one, you move the other. The eyepoint is always centered in the display screen, 1 y-unit farther from the view volume than the plot display screen.
- The z-axis (for the dependent variable) is displayed on the vertical axis; the x-axis (for the **INDEP** variable) is displayed on the horizontal axis; and the y-axis (for the **DEPND** variable) is the depth axis running from y_{near} to y_{far}.
- The plot display screen does not rotate in space, but remains parallel to the xz-plane and perpendicular to the y-axis, so you cannot get a top view of a function (looking down on the xy-plane) simply by moving the eyepoint/plot display screen. Instead you must transform the function so that the original dependent variable becomes one of the two independent variables. If it becomes the **DEPND** variable, you will be looking at the xy-plane; if it becomes the **INDEP** variable, you will be looking at the yz-plane.
- The display screen can't be *within* the view volume. The y_{near} coordinate must be at least 1 unit larger than y_e (the y-coordinate of the eyepoint).
- To visually center the plot in the display, make sure that x_e is set midway between x_{left} and x_{right} and that z_e is set midway between z_{low} and z_{high}.

Example: Plot the function $z = x^3 + y^3$, using a Wireframe plot with default settings for the view volume and eyepoint. Then repeat the plot, but change the eyepoint to a "high" vantage point—(0,-3, 5)—by adjust-ing the **ZE:** coordinate. Repeat again, restoring the default eyepoint, but viewing the *xy*-plane instead of the *xz*-plane.

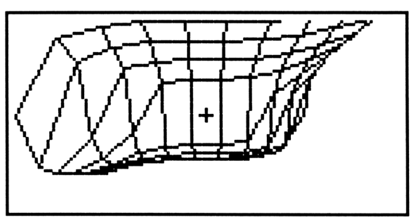
1. Purge VPAR, open the **PLOT** application, and put Wireframe in **TYPE:**: ['] [α] [α] [V] [P] [A] [R] [ENTER] [←] [PURG] [→] [PLOT] [▲] [α] [W] .
2. Put the function in **EQ:**: [▼] ['] [α] [←] [X] [y^x] [3] [+] [α] [←] [Y] [y^x] [3] [ENTER].
3. Put the two independent variable names into **INDEP:** and **DEPND:** [α] [←] [X] [ENTER] [►] [α] [←] [Y] [ENTER].



4. Now plot: **ERASE** **DRAW** ....
5. Next, return to the **PLOT OPTIONS** screen: [CANCEL] **OPTS**.



6. Change **ZE:** to 5 and redraw: [◄] [5] [ENTER] **OK** **ERASE** **DRAW** ....
7. To convert to an *xy*-view, you must solve the original function *equation* for *y* and use the result as the function, replacing *y* with *z* in **DEPND**. Solving for *y* yields $y = \sqrt[3]{x^3 - z}$.
8. At the **PLOT OPTIONS** screen, restore **ZE:** to 0 (which is its default): [CANCEL] **OPTS** [◄] [0] [ENTER].
9. Return to the main **PLOT** screen and enter the revised function in **EQ:**: [ENTER] ['] [→] [$\sqrt[x]{y}$] [3] [←] [,] [α] [←] [X] [y^x] [3] [−] [α] [←] [Z] [ENTER].
10. Enter the new name in **DEPND:** and then draw the plot: [▼] [α] [←] [Z] [ENTER] **ERASE** **DRAW** ....



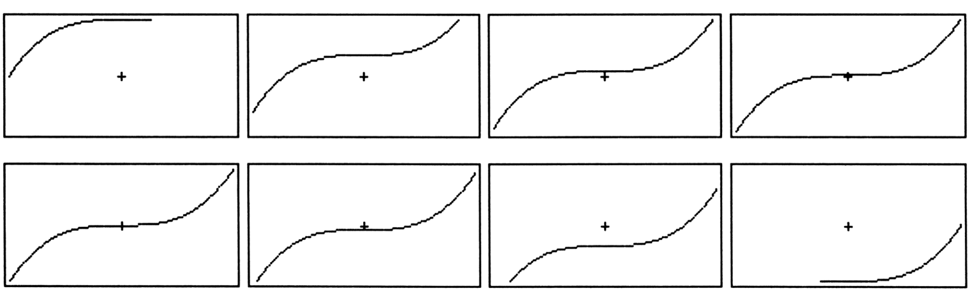
Y-Slice Plots

The Y-Slice plot draws a series of cross-sections of the three-dimensional surface, each perpendicular to the y-axis—one plot for each row in the sampling grid (unless you don't have enough unused memory left, in which case it draws as many of the rows as it can). Once it has finished drawing the "slices," it runs an animation loop of the slices until you press [CANCEL]. The animation allows you to visualize the surface by moving through it along the y-axis slice by slice.

If you wish to examine any of individual slices in greater detail after the original animation, you can check SAVE ANIMATION in the PLOT OPTIONS screen. Once you've finished viewing the animation and return to the stack you will find the slices stored as Graphic 131 x 64 on the stack with the number of slices stored on level 1. At this point, you may either rerun the animation by pressing [PRG] GROB [NXT] ANIM or view any one of the slices by storing it as the current PICTure (pressing [PRG] PICT PICT [STO] with the desired graphic on level 1) and viewing it with [PICTURE].

Example: Do a Y-Slice study of the function $z = x^3 + y^3$.

1. Return to the PLOT screen, set TYPE: to Y-Slice, reset the plot parameters, and enter the function in the EQ: field: [CANCEL] [▶] [α] [Y] [DEL] [▼] [ENTER] [▼] ['] [α] [↰] [X] [y^x] [3] [+] [α] [↰] [Y] [y^x] [3] [ENTER].
2. Enter the two independent variables and draw the Y-Slice plots: [α] [↰] [X] [ENTER] [▶] [α] [↰] [Y] [ENTER] ERASE DRAW. You will see each two-dimensional slice drawn, one after the other, until either the machine's memory gets low or one has been drawn for each row in the sampling grid (the number matches the value in STEPS: next to DEPND in the PLOT screen). The eight individual plots are:

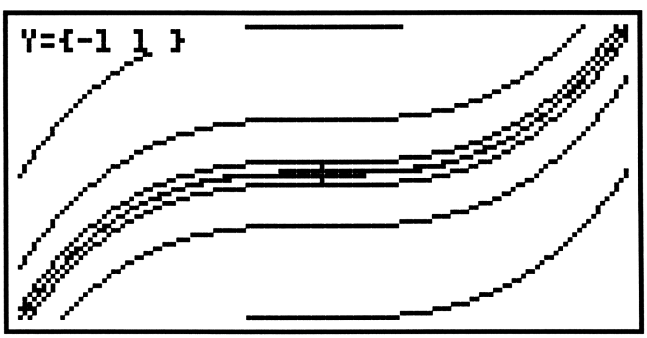


While the animated view of the "moving" slice through a surface may be preferable in some situations, you may want to see a composite picture of the slices instead in others.

The program YCOMP (see page 335) creates a single composite picture of a set of slices, labelling each slice as it's drawn. It takes the function from level 6, the list of independent variables from level 5, the x-view range (as a list) from level 4, the y-view range (as a list) from level 3, the z-view range (as a list) from level 2, and the number of slices to be drawn from level 1.

Example: Draw a composite Y-Slice plot of $z = x^3 + y^3$ using YCOMP. Use the default view volume and 8 slices; $-1 \le x \le 1$; $-1 \le y \le 1$; $-1 \le z \le 1$.

1. Return to the stack and enter the function: [CANCEL] [CANCEL] ['] [α] [←] [X] [y^x] [3] [+] [α] [←] [Y] [y^x] [3] [ENTER].
2. Enter the list of independent variables: [←] [{ }] [α] [←] [X] [SPC] [α] [←] [Y] [ENTER].
3. Enter the view volume ranges: [←] [{ }] [1] [+/−] [SPC] [1] [ENTER] [ENTER] [ENTER].
4. Enter the number of slices: [8] [ENTER].
5. Now draw the composite using YCOMP: [α] [α] [Y] [C] [O] [M] [P] [ENTER] or [VAR] (then [NXT] or [←] [PREV] as needed) [YCOM].

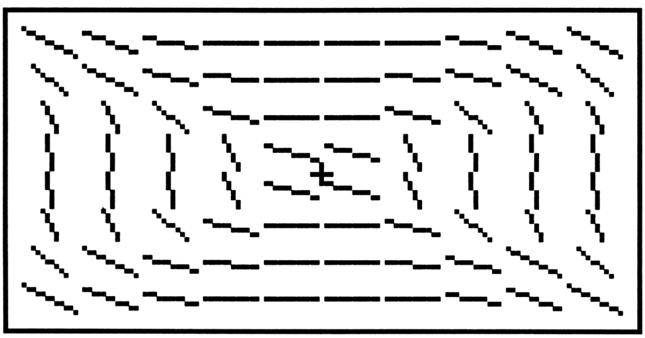


Contour Plots

The built-in Pseudo-Contour plot (Ps-Contour) transforms each point in the sampling grid into a short line segment that represents the slope of a contour of the function. Essentially, it first finds the implicit derivative of the given function and then plots its *slopefield*. The Ps-Contour plot then requires that you visually infer the actual contour curves from the lattice of tangent lines—a faster way to depict the contour curves than is plotting the contour lines themselves.

Example: Draw a Ps-Contour plot of the function $x^3 + y^3$, using default settings.

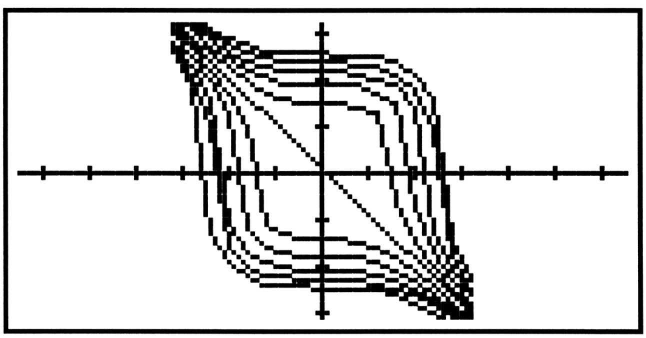
1. Return to the stack, open the PLOT application, set the TYPE: field to Ps-Contour, and reset the plot parameters: CANCEL ⤴ PLOT ▲ CHOOS ▲▲▲▲ ENTER DEL ▼ ENTER.
2. Enter the function into the EQ: field (if it's not already there): ▼ ' α ← X y^x 3 + α ← Y y^x 3 ENTER.
3. Enter the independent variables and draw the plot: α ← X ENTER ▶ α ← Y ENTER ERASE DRAW.



It is possible to plot true contour curves on the HP 48 as well, using a program. CONTOUR (see page 288) takes a two-variable function from level 4, a list of the independent variables from level 3, a list containing the range of contour values to be plotted from level 2, and the number of steps within the contour range from level 1, and plots a series of contour curves (one more than the level 1 input). If the number in level 1 is positive, CONTOUR erases the previous picture before plotting; if the number is negative it plots on top of the previous plot. It uses the current settings for X-LEFT, X-RIGHT, Y-NEAR, and Y-FAR to determine the display ranges (you may need to check or set these before using CONTOUR). It also uses the program SOLVPLT (see page 325).

Example: Use CONTOUR to plot contour curves for $z = x^3 + y^3$ and compare the result with the Ps-Contour plot you drew in the previous example.

1. At the stack, enter the function (it's in 'EQ'): [CANCEL] [CANCEL] [α] [α] [E] [Q] [ENTER].
2. The independent variables list: [←] [{ }] [α] [←] [X] [SPC] [α] [←] [Y] [ENTER].
3. Enter the range of z-values to use for drawing contour lines. While this choice is often a matter of thinking, experience, and trial and error, use a range of { –0.5 0.5 }: [←] [{ }] [•] [5] [+/–] [SPC] [•] [5] [ENTER].
4. Enter the number of contour intervals to use (the positive number to indicates that the previous plot be erased first): [1] [0] [ENTER].
5. Draw the contour plot: [α] [α] [C] [O] [N] [T] [O] [U] [R] [ENTER] or [VAR] (then [NXT] or [←] [PREV] as needed) CONT.



Note that CONTOUR plots curves that are evenly-spaced with respect to the dependent variable (z, in this case), while Ps-Contour plots are evenly-spaced with respect to the independent variables (in the sampling grid). Thus Ps-Contour gives an adequate idea of the *shape* of the contour curves, but it doesn't represent the steepness of the surface undulations very well.

Visualizing Two-Variable Vector Functions

To better understand how to visualize a two-variable (three-dimensional) vector function, recall how a single-variable (two-dimensional) vector function is represented by the HP 48: curves plotted with the Parametric plot type. This requires that you define two parametric functions—$x(t)$ and $y(t)$—combining them into a single complex function: $f(t) = x(t) + y(t)\boldsymbol{i}$.

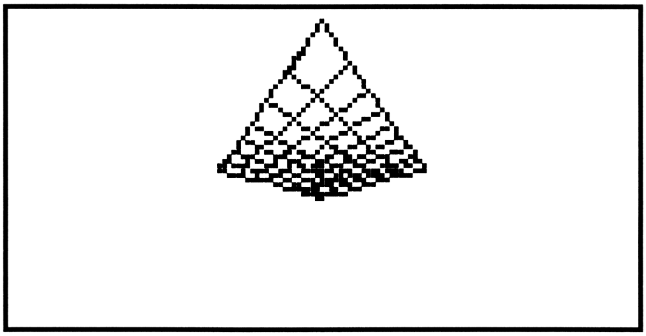
Unfortunately, complex numbers and complex functions can't handle more than two dimensions. So the Parametric Surface plot (Pr-Surface) uses a symbolic vector (list) of three parametric functions of two variables—$x(u,v)$, $y(u,v)$, and $z(u,v)$. If you let $y(u,v) = 0$ and allow $x(u,v)$ and $z(u,v)$ be functions of only one variable (either u or v, but not both) then the Parametric Surface plot simulates a two-dimensional Parametric plot as if it were viewed from a distant eyepoint.

The Parametric Surface plot uses points from the sampling grid—defined by the XX- and YY- ranges—as inputs for the symbolic vector function. The resulting points are then plotted, and those within the view volume are then displayed from the perspective of the given eyepoint. The XX-range (XX-LEFT to XX-RIGHT) and the YY-range (YY-NEAR to YY-FAR) determine the *plotting range* for the Parametric Surface plot; the X-, Y-, and Z- ranges determine the *display range* for the plot. Of course, the final plot displayed is the display range *as transformed by a perspective projection* from the eyepoint.

Example: Plot the surface described by $\mathbf{r}(u,v) = \{\ u+v \quad u-v \quad v-u\ \}$ for $-1 \le u \le 1$ and $-1 \le v \le 1$, as viewed from the default eyepoint.

1. Analyze the task. The parametrized surface and its plotting domain are given. To determine the display range (i.e. view volume), compute the minimum and maximum output values for each of the component functions. They are { { -2 2 } { -2 2 } { -2 2 } }.
2. Open the PLOT screen, set TYPE: to Pr-Surface, and reset parameters: (CANCEL) (→) (PLOT) (▲) CHOOS (▲) (ENTER) (DEL) (▼) (ENTER).
3. Enter the parametrization as a symbolic vector in the EQ: field: (▼) (←) ({ }) (') (α) (←) (U) (+) (α) (←) (V) (►) (') (α) (←) (U) (−) (α) (←) (V) (►) (') (α) (←) (V) (−) (α) (←) (U) (ENTER).

4. Enter the independent variables: [α][←][U][ENTER][▶][α][←][V][ENTER].
5. Enter plotting and display range (view volume): **OPTS** **XXYY** [1] [+/−][ENTER][1][ENTER][1][+/−][ENTER][1][ENTER][ENTER]; [2][+/−][ENTER] [2][ENTER][2][+/−][ENTER][2][ENTER][2][+/−][ENTER][2][ENTER][ENTER].
6. Draw the plot: **ERASE** **DRAW**.



Sometimes it helps to transform a scalar-valued function to a vector-valued one—a process known as *parametrization*—so that you can work with the function using vector-based tools, such as the Parametric Surface plot. Given a scalar function, $z = f(x,y)$, the easiest way to parametrize is to transfer the two independent variables into the symbolic vector unchanged and allow the third component to equal the function value. Thus, $z = f(x,y)$ becomes $\mathbf{r}(x,y) = \{\ x \quad y \quad f(x,y)\ \}$.

To visualize a parametrized surface using the HP 48, you must also consider the order of the components in the symbolic vector. The first component in the list is plotted on the horizontal axis, with a display range of X-LEFT to X-RIGHT. The second component is plotted along the implied axis of depth, with a range of Y-NEAR to Y-FAR. The third component is displayed along the vertical axis, with a range of Z-LOW to Z-HIGH. Note the pattern: { *horizontal depth vertical* }

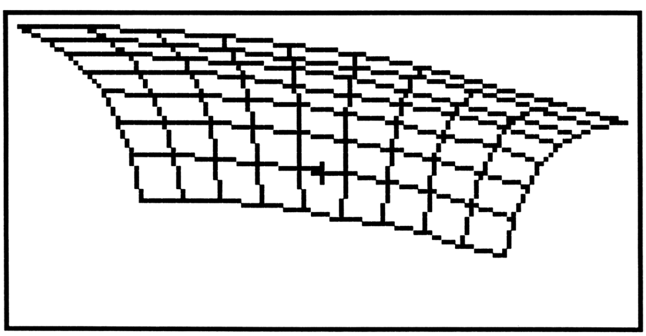
For example, to parametrize $x = z^2y^2$ and then plot it in conventional orientation (x horizontal and y vertical), set x as the dependent variable, y as INDEP, z as DEPND, and use the parametrization $\{\ z^2y^2 \quad z \quad y\ \}$. By contrast, to view the surface with the z-axis horizontal and x-axis vertical, use the same variables designations but a different parametrization: $\{\ z \quad y \quad z^2y^2\ \}$. Thus, the orientation of a Pr-Surface plot depends on the order of the components in the symbolic vector list, not on the order of the variable designations (unlike the other plot types).

The other important thing to do is to determine the appropriate plotting and display ranges. The most important item is the domain (i.e. plotting range) of the two independent variables. Once these are known, you can find reasonable display ranges for all three axes. Unless otherwise necessary, use an eyepoint whose horizontal and vertical coordinates (XE and ZE, respectively) are the midpoints of the horizontal and vertical display ranges (X-range and Z-range, respectively).

Following the previous example, suppose that for the function, $x = z^2y^2$, you let $0 \le z \le 1$ and $0 \le y \le 1$ be the plotting ranges (the XX-range is the plotting range for INDEP and the YY-range is the plotting range for DEPND). It's then easy to see that $0 \le x \le 1$ as well. Thus, in this case all three display ranges should be set to a low of 0 and a high of 1. The eyepoint can be set to (.5, -3, .5).

Example: Parametrize and plot $x^2 + y + z^3 = 10$, where $0 \le x \le 2$, $0 \le z \le 2$.

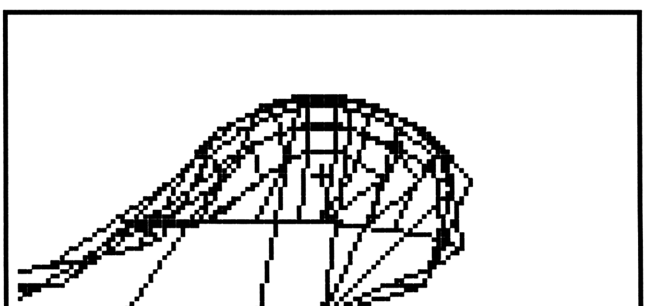
1. Analyze. The function is linear in y, so it's a good choice to be the dependent variable. Solving for y: $y = 10 - x^2 - z^3$. Parametrizing so that x is horizontal and y is vertical requires this symbolic vector: $\{x \;\; z \;\; 10 - x^2 - z^3\}$. Computing display ranges yields the following: { { 0 2 } { 0 2 } {-2 10 } }. Thus the eyepoint should be (1, -1, 4).
2. Enter the parametrization in EQ: [CANCEL] [▼] [▼] [←] [{ }] ['] [α] [←] [X] [▶] ['] [α] [←] [Z] [▶] ['] [1] [0] [−] [α] [←] [X] [y^x] [2] [−] [α] [←] [Z] [y^x] [3] [ENTER].
3. Enter the independent variables: [α] [←] [X] [ENTER] [▶] [α] [←] [Z] [ENTER].
4. Enter plotting and display ranges and eyepoint, then draw: OPTS XXYY [0] [ENTER] [2] [ENTER] [0] [ENTER] [2] [ENTER] [ENTER]; [0] [ENTER] [2] [ENTER] [0] [ENTER] [2] [ENTER] [2] [+/−] [ENTER] [1] [0] [ENTER]; [1] [ENTER] [1] [+/−] [ENTER] [4] [ENTER] [ENTER]; ERASE DRAW....



The final problem you may encounter when plotting parametrized surfaces is a domain that isn't rectangular—at least in the coordinate system you're using.

Example: Parametrize and plot the upper unit hemisphere of $z = \sqrt{1 - x^2 - y^2}$, where $x^2 + y^2 \leq 1$.

1. Analyze the task. The parametrization is { x y $\sqrt{1 - x^2 - y^2}$ }. The plotting ranges are $-1 \leq x \leq 1$ and $-1 \leq y \leq 1$. Note that although these ranges include possible points, such as (0.9, 0.7), that aren't in the function's domain, the points (1,0) and (0,-1) are in the domain. The display ranges are { {-1 1} {-1 1 } {0 1} } and the eyepoint should be (0, -2, .5).
2. Return to the **PLOT** screen, and enter the parametrization in the **EQ:** field: [CANCEL] [▼] [▼] [←] [{ }] ['] [α] [←] [X] [▶] ['] [α] [←] [Y] [▶] ['] [√x] [←] [( )] [1] [−] [α] [←] [X] [y^x] [2] [−] [α] [←] [Y] [y^x] [2] [ENTER].
3. Enter independent variables and plotting range: [▼] [α] [←] [Y] [ENTER] [OPTS] [XXRNG] [1] [+/−] [ENTER] [1] [ENTER] [1] [+/−] [ENTER] [1] [ENTER] [ENTER].
4. Enter the display range and eyepoint; draw the plot: [1] [+/−] [ENTER] [1] [ENTER] [1] [+/−] [ENTER] [1] [ENTER] [0] [ENTER] [1] [ENTER] [0] [ENTER] [2] [+/−] [ENTER] [•] [5] [ENTER] [ENTER] [ERASE] [DRAW].



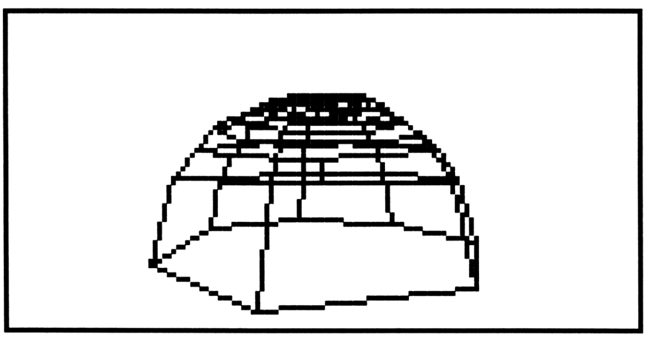
The plot of the hemisphere is distorted because Pr-Surface requires a rectangular plotting range (i.e. sampling grid), while the true domain in this case is *circular*. Thus, you are plotting points not in the true domain and thereby causing distortions in the output.

The solution is to use a parametrization that has a rectangular domain. This is where the polar coordinate systems—cylindrical and spherical—become useful.

For example, converting the function in the previous example to polar cylindrical coordinates will yield $z = \sqrt{1-r^2}$, $x = r\cos\theta$, $y = r\sin\theta$, and a parametrization of $\{\ r\cos\theta\ \ r\sin\theta\ \ \sqrt{1-r^2}\ \}$. Note that the domain of the new parametrization is now *rectangular*: $0 \le r \le 1$ and $0 \le \theta \le 2\pi$.

Example: Repeat the previous example using a cylindrical parametrization.

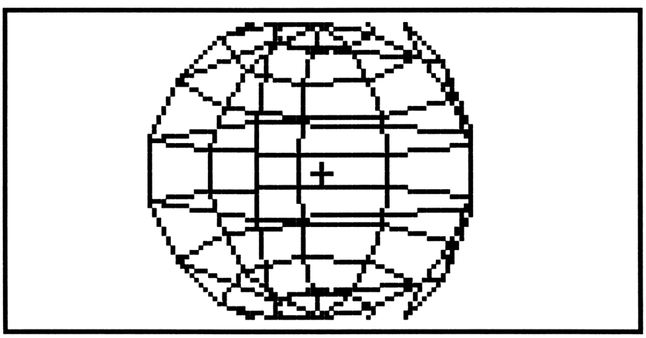
1. Analyze the task. The parametrization and plotting ranges are given above. The display ranges and eyepoint are the same as before.
2. Return to the **PLOT** screen, and enter the parametrization in the **EQ:** field and set ∡ to **Rad**: [CANCEL] [▼] ([+/−], if necessary) [▼] [←] [{ }] ['] [α] [←] [R] [×] [COS] [α] [→] [F] [▶] [▶] ['] [α] [←] [R] [×] [SIN] [α] [→] [F] [▶] [▶] ['] [√x] [←] [( )] [1] [−] [α] [←] [R] [yˣ] [2] [ENTER].
3. Enter the independent variables: [α] [←] [R] [ENTER] [▶] [α] [→] [F] [ENTER].
4. The plotting range: **OPTS** **XXRNG** [0] [ENTER] [1] [ENTER] [0] [ENTER] [6] [•] [2] [8] [ENTER] [ENTER].
5. Draw the plot: [ENTER] **ERASE** **DRAW**.



Of course, the hemisphere of the previous two examples can also be expressed in *spherical* coordinates which, not surprisingly, are perfectly suited to this function. Indeed, using spherical coordinates you not only have a rectangular domain, but you can plot the entire sphere because you no longer depend upon a square root.

Example: Plot a sphere using a spherical parametrization.

1. The spherical parametrization is { sinϕ cosθ sinϕsinθ cosϕ }. The plotting ranges are $0 \le \theta \le 2\pi$ and $0 \le \phi \le \pi$. The display ranges are { {-1 1} {-1 1 } {-.5 .5} }, and the eyepoint should be (0, -2, 0).
2. Return to the **PLOT** screen, and enter the parametrization in the **EQ:** field: [CANCEL] [▼] [▼] [←] [{ }] ['] [SIN] [α] [O] [α] [→] [9] [▶] [×] [COS] [α] [→] [F] [▶] [▶] ['] [SIN] [α] [O] [α] [→] [9] [▶] [×] [SIN] [α] [→] [F] [▶] [▶] ['] [COS] [α] [O] [α] [→] [9] [ENTER].
3. Enter independent variables, plotting and display ranges, and eyepoint: [α] [O] [α] [→] [9] [ENTER] **OPTS** **XXRNG** [0] [ENTER] [3] [•] [1] [4] [ENTER] [ENTER] [▼] [▼] [•] [5] [+/−] [ENTER] [•] [5] [ENTER] [▶] [▶] [0] [ENTER].
4. Draw the plot: **OK** **ERASE** **DRAW**.



Derivatives of Scalar Functions

The derivative of a single-variable function computes the local slope of the graph of the function. It also allows detection of extrema and determines the best linear approximation, the tangent, of the function at a given point. All of these properties can be extended to multivariable functions, but there are multiple derivatives.

Partial Derivatives

Partial derivatives are taken with respect to one of the multiple variables; the other variables are treated as constants. Partial derivatives compute the rate of change *along one of the axes* at a given point in the function. The built-in differentiation command in the HP48 is designed to compute partial derivatives. To use the built-in command, you should purge the variables first. If you use the program FDER (see page 294), you need not purge the variables; they will retain their values.

Example: Find the slope along the z-axis of $f(x,y,z) = \dfrac{z}{x+y}$ at (1,2,3).

1. The point: [1] ['] [α] [←] [X] [STO] [2] ['] [α] [←] [Y] [STO] [3] ['] [α] [←] [Z] [STO].
2. The function: ['] [α] [←] [Z] [÷] [←] [( )] [α] [←] [X] [+] [α] [←] [Y] [ENTER].
3. The differentiation variable: ['] [α] [←] [Z] [ENTER].
4. The partial derivative: [→] [∂]. Result: .333333333333

Example: Find all of the symbolic partial derivatives of e^{xyz}.

1. Enter the function and make two copies: ['] [←] [e^x] [α] [←] [X] [×] [α] [←] [Y] [×] [α] [←] [Z] [ENTER] [ENTER] [ENTER].
2. Gather the three copies into a list: [3] [PRG] [LIST] [→LIST].
3. Enter the independent variables in a list, make a copy and purge: [←] [{ }] [α] [←] [X] [SPC] [α] [←] [Y] [SPC] [α] [←] [Z] [ENTER] [ENTER] [←] [PURG].
4. Apply the derivative element-wise to the two lists: [→] [∂].
   Result: { 'y*z*EXP(x*y*z)' 'x*z*EXP(x*y*z)' 'x*y*EXP(x*y*z)' } This is { $\partial f/\partial x$ $\partial f/\partial y$ $\partial f/\partial z$ }.

Total Derivatives—Gradients

The total derivative is usually a compilation of partial derivatives—used to measure rates of change, detect relative extrema, and compute tangent planes to the whole function at a point. The most common version of the total derivative is the *gradient*, a vector of the partial derivatives of a function. Gathering together the partials into a symbolic vector list gives you the *gradient*, or total derivative. The program GRADI (see page 295) automates this. It takes the function from level 2 and a list of the independent variables from level 1.

Example: Use GRADI to find the gradient of $f(x,y,z) = x \sin^{-1} z - y \sin^{-1} z$.

1. Enter the function: ['] [α] [←] [X] [×] [←] [ASIN] [α] [←] [Z] [▶] [−] [α] [←] [Y] [×] [←] [ASIN] [α] [←] [Z] [ENTER].
2. Now the independent variables: [←] [{ }] [α] [←] [X] [SPC] [α] [←] [Y] [SPC] [α] [←] [Z] [ENTER].
3. Compute the gradient with GRADI: [α] [α] [G] [R] [A] [D] [I] [ENTER] or [VAR] GRADI. Result: `{ 'ASIN(z)' '-ASIN(z)' 'x/√(1-z^2)-y/√(1-z^2)' }`

Gradients are also used numerically, of course, to find the total rate of change at specific points on the function. In these cases, you will need to store the coordinates of the specific point in the correct variable names and evaluate the gradient.

Example: Find the value of the gradient of $f(x,y,z) = x^3y^4z^5$ at (1,-2,-1).

1. Enter the function: ['] [α] [←] [X] [y^x] [3] [×] [α] [←] [Y] [y^x] [4] [×] [α] [←] [Z] [y^x] [5] [ENTER].
2. Enter the list of independent variables: [←] [{ }] [α] [←] [X] [SPC] [α] [←] [Y] [SPC] [α] [←] [Z] [ENTER].
3. Enter the point as a list of coordinates; store in appropriate variables: [←] [{ }] [1] [SPC] [2] [+/−] [SPC] [1] [+/−] [ENTER] [←] [STACK] OVER [STO].
4. Find the gradient: [α] [α] [G] [R] [A] [D] [I] [ENTER] [1] [←] [« »] [←] [→NUM] [ENTER] [PRG] LIST PROC DOLIS. Result: `{ -48 32 80 }`

Finding the Plane Tangent to a Surface at a Point

Just as the tangent line is the best linear approximation to the curve of a single-variable function at a given point, so too is a tangent *plane* the best linear approximation to a surface represented by a two-variable function at a given point.

The equation of the tangent line or plane is the *first-order Taylor approximation* of the function at that point:

The tangent line to curve $f(x)$, at $x = a$, is $f(x) \approx f(a) + f'(a)(x - a)$

the tangent plane to surface $f(x,y)$, at (a,b), is $f(x,y) \approx f(a,b) + \textbf{grad}\, f \bullet \begin{bmatrix} x-a \\ y-b \end{bmatrix}$.

Note the analogies between the formulas: Instead of the single-variable derivative for the tangent line, the tangent plane uses the gradient vector. Instead of multiplying the derivative by the difference factor, the tangent plane uses the dot product of the gradient and a *vector* of difference factors—one for each variable.

And the use of vectors like this is easily extended to functions of any number of variables: $f(x_1, x_2, \ldots, x_n) \approx f(a_1, a_2, \ldots, a_n) + \textbf{grad}\, f \bullet \begin{bmatrix} x_1 - a_1 \\ x_2 - a_2 \\ \cdots \\ x_n - a_n \end{bmatrix}$

The program TPLAN (see page 331) computes the equation of the plane tangent to a scalar function at a given point. It takes the function from level 3, the list of variables from level 2, and the given point (as a list of coordinates) from level 1.

Example: Find the plane tangent to $f(x,y) = x^2 + 5xy - y^3$ at the point (-2,3).

1. Enter the function: ['] [α] [←] [X] [y^x] [2] [+] [5] [X] [α] [←] [X] [X] [α] [←] [Y] [−] [α] [←] [Y] [y^x] [3] [ENTER].
2. Enter the variables as a list: [←] [{ }] [α] [←] [X] [SPC] [α] [←] [Y] [ENTER].
3. Enter the point as a list of coordinates: [←] [{ }] [2] [+/−] [SPC] [3] [ENTER].
4. Find the tangent plane via TPLAN: [α] [α] [T] [P] [L] [A] [N] [ENTER] or [VAR] ([NXT] or [←] [PREV]) **TPLA**.

   Result: '80+11*x-37*y'

Example: Find the equation of the plane tangent to $f(x,y,z) = \ln(x^2 + y^2 + z^2)$ at the point (-2,1,5).

1. Enter the function: ['] [→] [LN] [α] [←] [X] [y^x] [2] [+] [α] [←] [Y] [y^x] [2] [+] [α] [←] [Z] [y^x] [2] [ENTER].
2. Enter the variables and the point as lists: [←] [{ }] [α] [←] [X] [SPC] [α] [←] [Y] [SPC] [α] [←] [Z] [ENTER] [←] [{ }] [2] [+/−] [SPC] [1] [SPC] [5] [ENTER].
3. Compute the tangent plane: [α] [α] [T] [P] [L] [A] [N] [ENTER] or [VAR] (then [NXT] or [←] [PREV] as needed) **TPLA** [←] [SYMBOLIC] [NXT] **→Q**.

   Result: '112809/80509-2/15*x+1/15*y+1/3*z'

Directional Derivatives

A function's partial derivatives find its rate of change in the directions of the coordinate axes. But you can move in an infinite number of directions, not just along an axis. The *directional derivative* is a generalization of the partial derivative that expresses the function's rate of change in *any* specified direction—found by taking the dot product of the gradient and the *unit vector* in the desired direction. (As an example of a unit vector: If the given direction vector is [2 -3 5], then its unit vector is $\frac{[2 \quad -3 \quad 5]}{\|[2 \quad -3 \quad 5]\|} \approx \frac{[2 \quad -3 \quad 5]}{6.1644} \approx [.32444 \quad -.48666 \quad 5.81111]$.)

Example: Find the derivative of $f(x,y) = x^2y + 2xy - y^3$ parallel to [3 4].

1. The function: ['][α][⬅][X][yˣ][2][×][α][⬅][Y][+][2][×][α][⬅][X][×][α][⬅][Y][−][α][⬅][Y][yˣ][3][ENTER].
2. Enter the variables, make a copy and purge: [⬅][{ }][α][⬅][X][SPC][α][⬅][Y][ENTER][ENTER][⬅][PURG].
3. Compute the gradient: [α][α][G][R][A][D][I][ENTER].
4. Enter the direction vector and find its unit vector, using symbolic vectors: [⬅][{ }][3][SPC][4][ENTER][ENTER][α][α][V][A][B][S][ENTER][÷].
5. The dot product, simplified: [α][α][S][D][O][T][ENTER][α][α][E][X][C][O][ENTER]. Result: `'.8*x^2+1.2*x*y-2.4*y^2+1.6*x+1.2*y'`

You can generalize the concept of the directional derivative even further: Instead of limiting the direction to a straight-line vector, you can find the derivative of a function with respect to a specified *curved path*. Expressing the curve as a one-variable vector function, $\mathbf{r}(t) = \{\ x(t) \quad y(t) \quad z(t)\}$, and the function as a scalar function of three variables, $f(x,y,z)$, you can then express the directional derivative of f along the path, $\mathbf{r}$, as $\frac{\text{grad}\, f(\mathbf{r}(t)) \bullet \mathbf{r}'(t)}{\|\mathbf{r}'(t)\|}$. The numerator of this expression is the rate of change of f along the path $\mathbf{r}$ at time t. Different parametrizations for $\mathbf{r}$ affect the rate of change differently. That is, although the path is the same for all parametrizations, the speed of its traversal is not. Thus, the denominator, the *speed* along $\mathbf{r}$, serves to eliminate the effect of the choice of parameterization.

The program DIRS (see page 293) computes the directional derivative of a scalar function along a parametrized curved path at a given point. It takes the function from level 4, the parametrized curve (a symbolic vector) from level 3, the list of function variables from level 2, and an equation defining the parameter variable and its given value (e.g 't=2') from level 1. DIRS then returns the symbolic directional derivative (level 2) and its numeric value at the given point (level 1).

Example: Find the directional derivative and rate of change at $t = 2$ for the function, $f(x,y,z) = x^3y^4z^5$ with respect to the path $\mathbf{r}(t) = \{\ t\ \ t^2\ t^3\ \}$.

1. The function: ['] [α] [←] [X] [y^x] [3] [×] [α] [←] [Y] [y^x] [4] [×] [α] [←] [Z] [y^x] [5] [ENTER].
2. The path, **r**: [←] [{ }] ['] [α] [←] [T] [▶] ['] [α] [←] [T] [y^x] [2] [▶] ['] [α] [←] [T] [y^x] [3] [ENTER].
3. The variables list: [←] [{ }] [α] [←] [X] [SPC] [α] [←] [Y] [SPC] [α] [←] [Z] [ENTER].
4. The equation of parameter-and-value: ['] [α] [←] [T] [←] [=] [2] [ENTER].
5. The directional derivative: [α] [α] [D] [I] [R] [→] [D] [ENTER] or [VAR] ([NXT] or [←] [PREV] as needed) **DIRS**. Result: 68755952.2903

   Thus, at $t = 2$, the function is increasing by nearly 69 million units for each unit travelled along **r**.

Of course, you can use DIRS to compute directional derivatives with respect to linear direction vectors, too, by expressing the linear direction parametrically....

Example: Use DIRS to find the directional derivative and rate of change for $f(x,y) = x^2y + 5xy - y^3$ in the direction of [3 4] at the point (-2 3).

1. Enter the function: ['] [α] [←] [X] [y^x] [2] [×] [α] [←] [Y] [+] [5] [×] [α] [←] [X] [×] [α] [←] [Y] [−] [α] [←] [Y] [y^x] [3] [ENTER].
2. For the path, **r**, convert the point and direction to a single-variable vector function representing a line: Enter the direction vector (symbolically), the parameter variable, then multiply; enter the point (as a list) and ADD. Thus: [←] [{ }] [3] [SPC] [4] [ENTER] ['] [α] [←] [T] [ENTER] [×] [←] [{ }] [2] [+/−] [SPC] [3] [ENTER] [MTH] **LIST** **ADD**.

3. Enter the list of variables: [←][{ }][α][←][X][SPC][α][←][Y] [ENTER].
4. Enter the equation defining the parameter and its value (set it to zero so that the desired point is selected): ['][α][←][T][←][=][0][ENTER].
5. Compute the directional derivative: [α][α][D][I][R][→][D] [ENTER] or [VAR] (then [NXT] or [←][PREV] as needed) [DIRD]. <u>Result</u>: -24.6

   Thus, as you move from (-2,3) in the direction of [3 4] the function is decreasing at the rate of 24.6 units per unit travelled.

The rate of increase in a given function is fastest in the direction of the gradient; and the rate of decrease is fastest in the direction opposite the gradient.

Example: A 500°K heat source radiates heat outward in a sphere. The temperature at a point (x,y,z) is $T(x,y,z)=\dfrac{500}{1+x^2+y^2+z^2}$.

What is the unit vector in the direction of fastest temperature increase at the point (1,2,3)? What rate of temperature increase is this?

1. Enter the function: ['][5][0][0][÷][←][( )][1][+][α][←][X][y^x][2][+][α][←][Y][y^x][2][+][α][←][Z][y^x][2][ENTER].
2. The variables list: [←][{ }][α][←][X][SPC][α][←][Y][SPC][α][←][Z][ENTER].
3. Store the point's coordinates in the appropriate variables: [←][{ }][1][SPC][2][SPC][3][ENTER][←][STACK] [OVER] [STO].
4. Compute the numerical gradient: [α][α][G][R][A][D][I][ENTER][1][« »][←][→NUM][ENTER][PRG] [LIST] [PROC] [DOT·].
5. Its magnitude: [ENTER][α][α][V][A][B][S][ENTER]. <u>Result</u>: 16.6295884
   This is (to 7 figures) the fastest rate of increase in temperature.
6. The unit vector: [÷]. <u>Result</u>: { -.2672612 -.5345225 -.8017837 }

   This direction of fastest increase makes sense: it is the same direction as { -1 -2 -3 } (which you can confirm by dividing the above result through by its first element)—and this is directly back to the heat-source origin from the point in question, (1,2,3).

Finding Critical and Stationary Points

Any point on a continuous, two-variable function, $z = f(x,y)$, whose gradient at that point is a zero vector is a *stationary critical point*. The "stationary" in the name stems from the function's behavior there: it's not changing in any direction. There is a *horizontal plane tangent to the surface* at a stationary critical point, whose equation is $z = k$, where k is the value of the function at the stationary critical point.

Example: Find the stationary critical points of the function $f(x,y) = x^2 + 5xy - y^3$ and the equations of the tangent planes at these points.

1. Enter the function: ['] [α] [←] [X] [y^x] [2] [+] [5] [×] [α] [←] [X] [×] [α] [←] [Y] [−] [α] [←] [Y] [y^x] [3] [ENTER].
2. Enter the list of variables, make two extra copies, moving one up the stack for later use and the other to purge: [←] [{ }] [α] [←] [X] [SPC] [α] [←] [Y] [ENTER] [ENTER] [ENTER] [4] [STACK] [ROLLD] [←] [PURG].
3. Compute the gradient: [α] [α] [G] [R] [A] [D] [I] [ENTER].

   Result: `{ '2*x+5*y' '5*x-3*y^2' }`
4. Swap the variable list into level 1 ([SWAP]) and estimate the values of x and y that will make each component of the gradient equal to zero. Obviously the point (0,0) works and is one critical point. But keep searching for others. The first component suggests three things:
   a. There is only one other critical point, since x and y are both linear in the first equation.
   b. x and y have opposite signs
   c. The x:y ratio in magnitude is approximately 5:2 .

   Looking at the second component suggests that while (5, -2) doesn't work very well (10, -4) gets fairly close. Thus, enter an initial guess of $x = 10$ and $y = -4$: [←] [{ }] [1] [0] [SPC] [4] [+/−] [ENTER].
5. Solve the two equations simultaneously via `NLSYS` (page 304) and rationalize: [α] [α] [N] [L] [S] [Y] [S] [ENTER] [5] [α] [α] [F] [I] [X] [ENTER] [←] [SYMBOLIC] [NXT] [→Q] [α] [α] [S] [T] [D] [ENTER].

   Result: `{ '125/12' '-(25/6)' }`

   Thus, the function has two critical points: (0,0) and $(\frac{125}{12}, -\frac{25}{6})$.

A stationary critical point can be any one of three things: a local *minimum*, a local *maximum*, or a *saddle point*. You can determine which situation applies by using a computation involving the *second-order partial derivatives* of the function: Arrange the second-order partial derivatives in a matrix, such as this one for a two-variable function, $\begin{bmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial y \partial x} & \frac{\partial^2 f}{\partial y^2} \end{bmatrix}$, and compute the determinant of the matrix. If it is less than zero, then the point is a saddle point. If it is greater than zero, then look at the value of $\frac{\partial^2 f}{\partial x^2}$. If that second-order partial is greater than zero, then the point is a local minimum; if less than zero, then the point is a local maximum. If the determinant of the matrix is zero, then the test fails to specify the nature of the stationary point.

The program `D2TST` (see page 290) automates this second-order partial derivatives test. It takes a function from level 3, the list of variables from level 2, and a list of stationary critical points (complex numbers) from level 1. It returns the list of points labeled with "Saddle," "Maximum," "Minimum," or "Inconclusive."

Example: Use the second-order derivative test to classify each of the stationary critical points you found in the previous example. Are they saddle points, local maxima, or local minima?

1. Enter the function: ['] [α] [←] [X] [y^x] [2] [+] [5] [×] [α] [←] [X] [×] [α] [←] [Y] [−] [α] [←] [Y] [y^x] [3] [ENTER].
2. Enter the list of variables: [←] [{ }] [α] [←] [X] [SPC] [α] [←] [Y] [ENTER].
3. Enter the list of stationary critical points (use decimal equivalents for the fractions): [←] [{ }] [←] [( )] [0] [←] [,] [0] [▶] [←] [( )] [1] [0] [•] [4] [1] [6] [6] [7] [←] [,] [4] [•] [1] [6] [6] [6] [7] [+/−] [ENTER].
4. Run the second-order partial derivatives test on the list of points: [α] [α] [D] [2] [T] [S] [T] [ENTER] or [VAR] (then [NXT] or [←] [PREV] as needed).

<u>Result</u>: `{ :Saddle: (0,0) :Minimum: (10.4166666667,-4.16666666667) }`

The Method of Lagrange Multipliers

Often the problem of finding function extrema of a function is complicated by additional *constraints* on the function inputs. However, the *method of Lagrange Multipliers* notes that, if an extremum exists for a differentiable function, *f*, subject to a constraint, *g*, then the gradient of *f* is a non-zero multiple of the gradient of *g*. That is, grad $f = \lambda$ grad g, where λ is a nonzero scalar, the Lagrange multiplier. Then, because the gradient is a *vector*, there is an equation for each element. For example, if *f* and *g* are functions of *x*, *y*, and *z*, then the Lagrange method gives the equations, $\frac{\partial f}{\partial x} = \lambda \frac{\partial g}{\partial x}$ $\frac{\partial f}{\partial y} = \lambda \frac{\partial g}{\partial y}$ $\frac{\partial f}{\partial z} = \lambda \frac{\partial g}{\partial z}$. With the original constraint, that's 4 equations in 4 unknowns (x, y, z, λ), a solvable system.

Example: On the plane $x - y + 2z = 5$, find the point closest to the origin.

1. The function minimized is the distance to the origin, $\sqrt{x^2 + y^2 + z^2}$, so $f(x,y,z) = x^2 + y^2 + z^2$. The constraint *g* is the plane, $x - y + 2z - 5$.
2. Enter the function *f* and its (purged) variables, then find its gradient: ['] [α] [↰] [X] [y^x] [2] [+] [α] [↰] [Y] [y^x] [2] [+] [α] [↰] [Z] [y^x] [2] [ENTER] [↰] [{ }] [α] [↰] [X] [SPC] [α] [↰] [Y] [SPC] [α] [↰] [Z] [ENTER] [ENTER] [↰] [PURG] [α] [α] [G] [R] [A] [D] [I] [ENTER]. <u>Result</u>: `{ '2*x' '2*y' '2*z' }`
3. Give constraint and variables; find its gradient: ['] [α] [↰] [X] [−] [α] [↰] [Y] [+] [2] [×] [α] [↰] [Z] [−] [5] [ENTER] [↰] [{ }] [α] [↰] [X] [SPC] [α] [↰] [Y] [SPC] [α] [↰] [Z] [ENTER] [α] [α] [G] [R] [A] [D] [I] [ENTER]. <u>Result</u>: `{ 1 -1 2 }`
4. Multiply by λ and subtract from the previous one: ['] [α] [↱] [L] [ENTER] [×] [−]. <u>Result</u>: `{ '2*x-λ' '2*y+λ' '2*z-2*λ' }`
5. Enter the constraint function and append it to the list: ['] [α] [↰] [X] [−] [α] [↰] [Y] [+] [2] [×] [α] [↰] [Z] [−] [5] [ENTER] [+].
6. Enter the variables and initial guesses ({ 1 -1 1 2} seems OK). Use `NLSYS` to solve, then rationalize: [↰] [{ }] [α] [↰] [X] [SPC] [α] [↰] [Y] [SPC] [α] [↰] [Z] [SPC] [α] [↱] [L] [ENTER] [↰] [{ }] [1] [SPC] [1] [+/−] [SPC] [1] [SPC] [2] [ENTER] [α] [α] [N] [L] [S] [Y] [S] [ENTER] [8] [α] [α] [F] [I] [X] [ENTER] [↰] [SYMBOLIC] [NXT] [→Q]. <u>Result</u>: `{ '5/6' '-(5/6)' '5/3' '5/3' }`

   The solution point is $(\frac{5}{6}, -\frac{5}{6}, \frac{5}{3})$. (Just discard λ, the 4th element.)

Derivatives of Vector Functions

Before looking at multivariable vector functions, consider single-variable vector functions. Expressing a function parametrically, say, $\mathbf{r}(t) = \{\ x(t)\quad y(t)\quad z(t)\ \}$, transforms a single function into a *list* of functions. Finding the derivative of such a list is nothing more than finding the derivative of each element of the list (i.e. each component of the symbolic vector): $\mathbf{r}'(t) = \dfrac{\partial \mathbf{r}}{\partial t} = \left\{ \dfrac{\partial x}{\partial t} \quad \dfrac{\partial y}{\partial t} \quad \dfrac{\partial z}{\partial t} \right\}$

The program VDER (see page 332) computes the derivative of vector function with respect to one variable. It takes the symbolic vector from level 2 and the differentiation variable from level 1, returning the derivative vector function.

Example: Find the derivative of $\mathbf{r}(t) = \{\ t\quad t^2\quad t^3\ \}$.

1. Enter the vector function: [←][{ }][α][←][T]['][α][←][T][y^x][2][▶]['][α][←][T][y^x][3][ENTER].
2. Enter the differentiation variable: ['][α][←][T][ENTER].
3. Compute the derivative using VDER: [α][α][V][D][E][R][ENTER] or [VAR] (then [NXT] or [←][PREV] as needed) [VDER].

   Result (in STD mode): `{ 1 '2*t' '3*t^2' }`

To find higher-order derivatives of vector functions, you need only to use VDER more than once.

Example: Find the second derivative of $\mathbf{r}(t) = \{\ t\ \ t^2\ \ t^3\ \}$. Use the previous result as a starting point.

1. Using the previous result as the vector function, enter the differentiation variable: ['][α][←][T][ENTER].
2. Compute the derivative using VDER: [α][α][V][D][E][R][ENTER] or [VAR] (then [NXT] or [←][PREV] as needed) [VDER].

   Result: `{ 0 2 '6*t' }`

One good application of the vector-valued functions is computing the *curvature* —the rate of change of the direction of the tangent— of a curve at a point on the curve. Without going into the derivation of the formula, the curvature of a curve defined parametrically at given point t is $\kappa = \dfrac{\|\mathbf{r}''(t) \times \mathbf{r}'(t)\|}{\|\mathbf{r}'(t)\|^3}$.

The *radius of curvature*—the radius of the circle whose curvature matches that of the curve—is the reciprocal of the curvature. That is, if the curvature is 0.2, then the radius is 5.

The program CURV (see page 289) takes the symbolic vector form of the curve from level 3, the parameter variable from level 2, and the value of the parameter at the desired point from level 1 and returns the numeric curvature to level 1.

Example: Find the curvature and radius of curvature for the curve defined by

$$\mathbf{r}(t) = \{\ 1 - t \quad t^2 + 1 \quad 2t^3/3 + 1\ \},\ \text{at } t = \tfrac{1}{2}.$$

1. Enter the vector function: [↰][{ }]['][1][−][α][↰][T][▶]['][α][↰][T][yˣ][2][+][1][▶]['][2][×][α][↰][T][yˣ][3][÷][3][+][1][ENTER].
2. Enter the parameter variable and its value: ['][α][↰][T][ENTER][•][5][ENTER].
3. Compute the curvature using CURV, then rationalize the result: [α][α][C][U][R][V][ENTER] or [VAR] ([NXT] or [↰][PREV] as needed) CURV [↰][SYMBOLIC][NXT] →Q.

Result: '8/9'

Of course, the concepts of single-variable vector derivatives can be extended (to a substantial degree) to derivatives of multivariable vector functions....

Partial Derivatives

Just as the built-in ∂ command can take partial derivatives for multivariable scalar functions, so VDER can take partial derivatives for multivariable vector functions. (Scalar functions have scalar partials; vector functions have vector partials.)

Example: Compute $\dfrac{\partial r}{\partial y}$ for $\mathbf{r}(x,y,z) = \{\ x^2y\ \ xyz^2\ \ y^2z^3\ \}$.

1. Enter **r**: [←][{ }]['][α][←][X][yˣ][2][×][α][←][Y][▶]['][α][←][X][×][α][←][Y][×][α][←][Z][yˣ][2][▶]['][α][←][Y][yˣ][2][×][α][←][Z][yˣ][3][ENTER].
2. Specify y, then differentiate via VDER: ['][α][←][Y][ENTER][α][α][V][D][E][R][ENTER]. Result: `{ 'x^2' 'x*z^2' '2*y*z^3' }`

Total Derivative—Jacobian

The total derivative for a scalar function is the gradient—a *vector.* The total derivative for a vector function is the *Jacobian*—a *matrix*. Each row of the Jacobian matrix is a gradient of one of the vector function's components.

The program JACOB (page 299) takes a symbolic vector function from level 2 of the stack and a list of the variables from level 1 and returns the Jacobian matrix to the stack. Remember to purge the variables before using JACOB.

Example: Find the total derivative, **F**′, at (3, 2, –3), if $\mathbf{F}(x,y,z) = \{3x^2\, 4y^2\, 5z^2\ \}$

1. Enter the symbolic vector: [←][{ }]['][3][×][α][←][X][yˣ][2][▶]['][4][×][α][←][Y][yˣ][2][▶]['][5][×][α][←][Z][yˣ][2][ENTER].
2. Enter the list of variables, make an extra copy and purge: [←][{ }][α][←][X][SPC][α][←][Y][SPC][α][←][Z][ENTER][ENTER][←][PURG].
3. Compute the Jacobian matrix: [α][α][J][A][C][O][B][ENTER] or [VAR] (then [NXT] or [←][PREV] as needed) [JACOB].

Result:
```
{ { '3*(2*x)' 0 0 }
  { 0 '4*(2*y)' 0 }
  { 0 0 '5*(2*z)' } }
```

4. Store the point's coordinates in the appropriate variables and compute the Jacobian matrix: [←][{ }][3][SPC][2][SPC][3][+/−][ENTER][←][{ }][α][←][X][SPC][α][←][Y][SPC][α][←][Z][ENTER][STO][1][« »][1][« »][→NUM][▼][PRG] LIST PROC DOLIS [▼] DOLIS [ENTER].

   Result: { { 18 0 0 }
   { 0 16 0 }
   { 0 0 -30 } }

Example: Find the symbolic total derivative ($\mathbf{P}'$) if

$$\mathbf{P}(x,y,z) = \{\ x + y + z \quad x + y^2 + z^3 \quad x^3 - y^2 - z\ \}.$$

1. Enter the symbolic vector: [←][{ }]['][α][←][X][+][α][←][Y][+][α][←][Z][▶]['][α][←][X][+][α][←][Y][y^x][2][+][α][←][Z][y^x][3][▶]['][α][←][X][y^x][3][−][α][←][Y][y^x][2][−][α][←][Z][ENTER].
2. Enter the list of variables, make an extra copy and purge: [←][{ }][α][←][X][SPC][α][←][Y][SPC][α][←][Z][ENTER][ENTER][←][PURG].
3. Compute the Jacobian matrix: [α][α][J][A][C][O][B][ENTER].

   Result: { { 1 1 1 }
   { 1 '2*y' '3*z^2' }
   { '3*x^2' '-(2*y)' -1 } }

The determinant of the Jacobian of a three-component vector function of three variables is the *triple scalar product* of the gradients of its components.

That is, for $\mathbf{f}(x,y,z) = \{\ P(x,y,z)\ \ Q(x,y,z)\ R(x,y,z)\ \}$,

$$\left|\mathbf{f}'(x,y,z)\right| = \begin{vmatrix} \frac{\partial P}{\partial x} & \frac{\partial P}{\partial y} & \frac{\partial P}{\partial z} \\ \frac{\partial Q}{\partial x} & \frac{\partial Q}{\partial y} & \frac{\partial Q}{\partial z} \\ \frac{\partial R}{\partial x} & \frac{\partial R}{\partial y} & \frac{\partial R}{\partial z} \end{vmatrix} = \text{grad}\, P \bullet \left(\text{grad}\, Q \times \text{grad}\, R\right).$$

As an arbitrarily-sized "box-element" is transformed by a vector function, its volume will increase by a scale factor equal to the absolute value of the determinant of the Jacobian. This scale factor plays a role during a change of variables when integrating, as you'll see later in this chapter (see pages 262 and 266).

Divergence, Curl, and Laplacian

In addition to the Jacobian, vector functions have three special kinds of "total" derivatives, each of which tells you something different about the function. The three types are:

- The *divergence* of a vector function is a scalar value representing the rate of expansion or compression of the function. If you choose a point from a given surface and let it be the center of a tiny cube, the divergence at that point is the rate of change of the *volume* of the cube. Divergence can be computed from the Jacobian matrix—it is the *trace* (the sum of its diagonal elements) of the Jacobian. The vector function is said to be *incompressible* at a point if the divergence is zero.
- The *curl* of a vector function is a vector representing the direction of the axis of maximum rotational change for the vector function. If you choose a point from a given surface and let it be the center of a tiny cube, the curl at that point is the axis of rotation of that cube and the *magnitude* of the curl is exactly twice the angular velocity of the cube. A vector function is said to be *irrotational* at a point if the curl is the zero vector.
- The *laplacian* is a measure of the difference between the *average value* of a function near a given point and the *actual value* of the function at the point. It roughly describes a rate of change for the average value of a function in the neighborhood of a given point. The laplacian is defined both for scalar and for vector functions. For scalar functions, the laplacian is a scalar—the divergence of the gradient. For vector functions, the laplacian is vector in which each component is the divergence of the gradient of its corresponding component in the original function.

Each of these three types of derivative has its own program—VDIV (page 333), CURL (page 289), and LAPLC (page 299). All three programs take the same set of inputs—the symbolic vector function from level 2 and the list of variables from level 1. There should be the same number of variables in the level-1 list as there are components in the level-2 vector. If you want the symbolic result, you must purge the variables in the level-1 list before using a program. If you want the numeric result, store values in the level-1 variables before using a program.

Look at some examples:

Example: Find the symbolic divergence of $\mathbf{F}(x,y,z) = \{3x^2\ 4y^2\ 5z^2\}$. Is the function expanding or compressing at the point (3,2,-3)?

1. Enter the symbolic vector: [←][{ }]['][3][×][α][←][X][y^x][2][▶]['][4][×][α][←][Y][y^x][2][▶]['][5][×][α][←][Z][y^x][2][ENTER].
2. Enter the list of variables, make an extra copy and purge: [←][{ }][α][←][X][SPC][α][←][Y][SPC][α][←][Z][ENTER][ENTER][←][PURG].
3. Compute the divergence: [α][α][V][D][I][V][ENTER] or [VAR] (then [NXT] or [←][PREV] as needed) VDIV. Result: '6*x+8*y+10*z'
4. Store the point coordinates in appropriate variables and evaluate the divergence: [←][{ }][3][SPC][2][SPC][3][+/-][ENTER][←][{ }][α][←][X][SPC][α][←][Y][SPC][α][←][Z][ENTER][STO][←][→NUM]. Result: 4

   A positive divergence indicates expansion at (3,2,-3).

Example: Find the curl of $\mathbf{F}(x,y,z) = \{\ 3x^2\ 4y^2\ 5z^2\ \}$.

1. Enter the symbolic vector: [←][{ }]['][3][×][α][←][X][y^x][2][▶]['][4][×][α][←][Y][y^x][2][▶]['][5][×][α][←][Z][y^x][2][ENTER].
2. Enter the list of variables, make an extra copy and purge: [←][{ }][α][←][X][SPC][α][←][Y][SPC][α][←][Z][ENTER][ENTER][←][PURG].
3. Find the curl: [α][α][C][U][R][L][ENTER] or [VAR] ([NXT] or [←][PREV]) CURL. Result: { 0 0 0 }. The function is *irrotational*.

Example: Find the laplacian of the function, $\mathbf{F}(x,y,z) = \{3x^2\ 4y^2\ 5z^2\}$.

1. Enter the symbolic vector: [←][{ }]['][3][×][α][←][X][y^x][2][▶]['][4][×][α][←][Y][y^x][2][▶]['][5][×][α][←][Z][y^x][2][ENTER].
2. Enter the list of variables, make an extra copy and purge: [←][{ }][α][←][X][SPC][α][←][Y][SPC][α][←][Z][ENTER][ENTER][←][PURG].
3. Compute the laplacian: [α][α][L][A][P][L][C][ENTER] or [VAR] (then [NXT] or [←][PREV] as needed) LAPLC. Result: { 6 8 10 }

Tangent Planes to Parametrized Surfaces

To find the equation of the plane tangent to a parametrized surface at a given point, you must first compute the *fundamental vector product (FVP)*—the cross product of the two partial derivatives of the surface. If $\mathbf{r}(u,v) = \{\ x(u,v)\quad y(u,v)\quad z(u,v)\ \}$ is a parametrized surface, then FVP $\mathbf{r}(u,v) = \dfrac{\partial \mathbf{r}}{\partial u} \times \dfrac{\partial \mathbf{r}}{\partial v}$.

Note that $\partial\mathbf{r}/\partial$u and $\partial\mathbf{r}/\partial$v both lie in the plane tangent to the surface at a given point. Thus their cross product is *normal* to the tangent plane and can thereby be used to determine the plane's equation. (Recall that if you know a particular point in a plane (x_0,y_0,z_0) and a vector $\{\ A\ B\ C\}$ normal to the plane, then the equation of the plane is: $A(x - x_0) + B(y - y_0) + C(z - z_0) = 0$.)

The program FVP (page 294) automates the computation of the fundamental vector product at a given point on a given parametrized surface. FVP takes the symbolic vector function from level 3, a list of the two independent variables from level 2, and a list containing the two coordinates for the independent variables at the given point. It returns the numeric fundamental vector product as a symbolic vector (i.e. in a list) to level 1.

Example: Find an equation of the plane tangent to $\mathbf{S}(u,v) = \{\ u\quad v\quad u^2v^2\ \}$ at the point (-1,2,4).

1. Enter the vector function for the parametrized surface: [←][{ }][α][←][U][SPC][α][←][V][SPC]['][α][←][U][y^x][2][×][α][←][V][y^x][2][ENTER].
2. Enter the list of the variable names and a list of their values: [←][{ }][α][←][U][SPC][α][←][V][ENTER][←][{ }][1][+/−][SPC][2][ENTER].
3. Compute the fundamental vector product: [α][α][F][V][P][ENTER] or [VAR] (then [NXT] or [←][PREV] as needed) [FVP].
4. Assemble an equation for the plane and then simplify: [←][{ }][α][←][X][SPC][α][←][Y][SPC][α][←][Z][ENTER][←][{ }][1][+/−][SPC][2][SPC][4][ENTER][−][×][MTH][LIST][ELIST][α][α][E][X][C][O][ENTER].

<u>Result</u>: `'12+8*x-4*y+z'`

Double Integration with Scalar Functions

The definite integral of a function of one variable, geometrically interpreted, is related to *area*—the sum of a series of "strips" that have height but whose width approaches zero. By extension, the definite integral of a function of two variables, geometrically interpreted, is related to *volume*—the sum of a series of flat "plates" that have height and width but whose thickness approaches zero.

The definite integral of a function of two variables, $f(x,y)$, is called a *double integral*, and represents the net volume between $f(x,y)$ and the xy-plane, over a defined region R in the xy-plane. The volume is *net* volume in that volume: above the xy-plane is counted as positive and volume below the xy-plane as negative.

The general form of the double integral is $\iint_R f(x,y)\,dA$. The dA refers to a summing element that is two-dimensional (area). The HP 48, however, cannot use the general form of the double integral. It can use only single definite integrals with specified upper and lower limits, so it requires that you transform the general form into a *iterated* form—a single integral whose integrand is itself an integral expression, such as $\int_a^b \left(\int_c^d f(x,y)\,dy \right) dx$. For each x between a and b, a vertical cross-section of R runs from $y = c$ to $y = d$.

To better visualize a double integral, think of the volume it computes as that of a book sitting upright on a shelf, with its binding facing you. Now modify your image so that the area of the book's contact with the shelf isn't necessarily a rectangle. The top of the book isn't necessarily flat, either.

The pages of the book represent slices of the volume between two limits (the covers of the book). The double integral (volume of the book) represents the sum of the areas of the slices (pages) between the lower limit (back cover) and the upper limit (front cover); the inner integral in a double integral computes the area of a slice—an area that may be constant (all pages the same size) or may instead be a function of its position within the volume (book). In either case, the outer integral totals the areas of the slices (pages) into a volume (book).

Using the iterated form of the double integral requires that you define the region R so as to establish values for the four limits a, b, c, and d. Now, each slice (page) intersects R in a line segment. If the line segments for the slices have varying lengths, then c and d are probably functions of x: $c = g(x)$ and $d = h(x)$. The limits of a and b are then the minimum and maximum x-values (front and back cover positions), respectively, in R. The iterated form becomes $\displaystyle\int_{\min R_x}^{\max R_x}\left(\int_{g(x)}^{h(x)} f(x,y)\,dy\right)dx$.

Of course, you could also find the volume of a object by slicing it horizontally instead of vertically, if it would be more convenient to do so—such as if the book in the foregoing analogy were sitting on the shelf with its front cover showing. Then the iterated form becomes $\displaystyle\int_{\min R_y}^{\max R_y}\left(\int_{g(y)}^{h(y)} f(x,y)\,dx\right)dy$.

Using the book analogy, the front cover is now the minimum y-value of R, the back cover the maximum y-value of R, and the area of the pages (slices) now depend on their y-position in the book.

It's important to execute the double integral correctly on your HP 48, or the computation will take seemingly forever. Here are some do's and don't's:

Don't: Create a symbolic nested double integral expression and then press [→NUM]. The HP 48 will take too long to evaluate it (unless perhaps the display/precision is set to 2 or 3 places).

Do: Compute the inner integral first, evaluating and symbolically simplifying as much as possible, then use the result as the integrand of the outer integral which can then be finished with [→NUM].

Maybe: Create a symbolic nested integral expression and use MLTINT (page 303) to evaluate it efficiently (starting from the inside-out).

Remember: It is always better to evaluate the inner integral symbolically, if at all possible. Note that if the inner integrand cannot be evaluated symbolically by the HP 48, then you may either have to work it manually before using the HP 48 for the outer integral or resort to numerical double integration (i.e. the long "Wrong" way) using a small display/precision setting. Look at some examples:

Example: Find the volume between the graph of $z = x^2y^3$ and the xy-plane, over a rectangular region with vertices (-1,1), (-1, 4), (3,4), and (3,1).

1. Since $-1 \le x \le 3$ and $1 \le y \le 4$, the double integral is $\int_1^4 \int_{-1}^3 x^2 y^3 \, dx \, dy$.

   The polynomial integrand should readily evaluate symbolically.
2. Enter the "outer" limits, then the "inner" limits: [1] [ENTER] [4] [ENTER]; [1] [+/–] [ENTER] [3] [ENTER].
3. Enter the integrand and "inner" variable of integration: ['] [α] [←] [X] [y^x] [2] [×] [α] [←] [Y] [y^x] [3] [ENTER] ['] [α] [←] [X] [ENTER].
4. Integrate: [→] [∫] [EVAL]. <u>Result</u>: '9*y^3+.333333333333*y^3'
5. Enter the "outer" variable of integration; do the outer integration to find the volume: ['] [α] [←] [Y] [ENTER] [→] [∫] [EVAL]. <u>Result</u>: 595

Example: Find the volume between the graph of $z = x^2 + xy - y^2$ and the xy-plane over a region bounded by the graph of $y = x^2$, the x-axis (y =0), the y-axis, and the line x=1.

1. This time the region of integration isn't rectangular. Since $0 \le x \le 1$ and $0 \le y \le x^2$, the double integral is $\int_0^1 \int_0^{x^2} x^2 + xy - y^2 \, dy \, dx$.
2. Enter the "outer" limits, then the "inner" limits: [0] [ENTER] [1] [ENTER]; [0] [ENTER] ['] [α] [←] [X] [y^x] [2] [ENTER].
3. The integrand and "inner" variable of integration: ['] [α] [←] [X] [y^x] [2] [+] [α] [←] [X] [×] [α] [←] [Y] [–] [α] [←] [Y] [y^x] [2] [ENTER] ['] [α] [←] [Y] [ENTER].
4. Evaluate the "inner" integral: [→] [∫] [EVAL] [←] [SYMBOLIC] [COLCT].

   <u>Result</u>: '.5*x^2^2*x+x^4-.333333333333*x^2^3'
5. Enter the "outer" variable of integration; integrate to get the volume: ['] [α] [←] [X] [ENTER] [→] [∫] [←] [→NUM]. <u>Result</u>: .235714285714

   Converting to a fraction via →Q or →Qπ yields '33/140'.

Example: Repeat the previous example using MLTINT.

1. Enter the symbolic nested double integral: ['] [→] [∫] [0] [←] [,] [1] [←] [,] [→] [∫] [0] [←] [,] [α] [←] [X] [y^x] [2] [←] [,] [α] [←] [X] [y^x] [2] [+] [α] [←] [X] [×] [α] [←] [Y] [−] [α] [←] [Y] [y^x] [2] [←] [,] [α] [←] [Y] [▶] [←] [,] [α] [←] [X] [ENTER].
2. Evaluate the double integral: [α] [α] [M] [L] [T] [I] [N] [T] [ENTER] or [VAR] ([NXT] or [←] [PREV]) MLTIN. Result: .235714285714.

Example: Compute $\int_0^2 \int_{-1}^3 \frac{x^2}{4+y^2}\, dy\, dx$.

1. Enter the "outer" limits: [0] [ENTER] [2] [ENTER].
2. Enter the "inner" limits: [1] [+/−] [ENTER] [3] [ENTER].
3. Enter the integrand and "inner" variable of integration: ['] [α] [←] [X] [y^x] [2] [÷] [←] [( )] [4] [+] [α] [←] [Y] [y^x] [2] [ENTER] ['] [α] [←] [Y] [ENTER].
4. Integrate: [→] [∫] [EVAL]. Result: '∫(-1,3,x^2/(4+y^2),y)'

   Hmm… this integral didn't evaluate symbolically. Your options:

   a. Symbolically evaluate this inner integral manually, then proceed with the computation using the HP 48.

   b. Symbolically evaluate this inner integral by using XINT, hoping that IPATS has the relevant matching pattern; or by using a combination of the other symbolic integration strategies discussed in chapter 4. Then do the outer integral as usual.

   c. With low precision for quicker computation, numerically estimate the complete double integral, using either [←] [→NUM] or MLTINT (which offers at least a four-place numerical approximation).

5. If you chose options a or b, the inner integral expression will have been replaced with its antiderivative. If you chose option c, the inner integral expression remains unevaluated. Enter the "outer" variable of integration and integrate: ['] [α] [←] [X] [ENTER] [→] [∫].
6. If you chose options a or b, then evaluate the integral (either [EVAL] [EVAL] or [←] [→NUM]). If you chose option c, then evaluate the integral numerically to four decimal places using MLTINT ([α] [α] [M] [L] [T] [I] [N] [T] [ENTER]). Result (to 4 places): 1.9285

Segmenting Double Integrals

Some regions of integration you'll encounter have *vertices* other than at the outer integral's limits. When this occurs, you must subdivide (or *segment*) the double integral into two or more double integrals. The limits for both inner and outer integrals are usually affected by segmentation.

Example: Let R be the triangular region with vertices at (0,0), (1,3), and (2,2).

Find $\iint_R e^{x-y}\,dy\,dx$ by using segmentation.

1. First, study R. The x-range is $0 \le x \le 2$, and in that range (at $x = 1$) lies a vertex. Before the vertex, the upper limit for y is the line connecting (0,0) and (1,3), or $y = 3x$. After the vertex, the upper limit for y is the line connecting (1,3) and (2,2) or $y = 4 - x$. Thus, it makes sense to split the double integral around $x = 1$. The first subregion is $0 \le x \le 1$ and $x \le y \le 3x$; the other is $1 \le x \le 2$ and $x \le y \le 4 - x$.
2. Enter the segmented integral, $\int_0^1 \int_x^{3x} e^{x-y}\,dy\,dx + \int_1^2 \int_x^{4-x} e^{x-y}\,dy\,dx$, onto the stack: Press [⬅] [EQUATION] [➡] [∫] [0] [▶] [1] [▶] [➡] [∫] [α] [⬅] [X] [▶] [3] [α] [⬅] [X] [▶] [⬅] [eˣ] [α] [⬅] [X] [−] [α] [⬅] [Y] [▶] [▶] [α] [⬅] [Y] [SPC] [α] [⬅] [X] [+] [➡] [∫] [1] [▶] [2] [▶] [➡] [∫] [α] [⬅] [X] [▶] [4] [−] [α] [⬅] [X] [▶] [⬅] [eˣ] [α] [⬅] [X] [−] [α] [⬅] [Y] [▶] [▶] [α] [⬅] [Y] [SPC] [α] [⬅] [X] [ENTER].
3. Numerically compute the segmented integral using `MLTINT`: [VAR] [MLTI]. Result: `1.13533528324`

   This matches the analytical result ($1 + e^{-2}$) to full precision.

Double Integrals with Polar Coordinates

It's often more convenient to use polar functions (of two variables, r and θ) to perform a double integration, particularly if the region R is more simply described by polar functions. If the problem is already given entirely in polar form, then computing the double integral is no different than for rectangular functions.

Example: Compute $\int_{\frac{\pi}{4}}^{\frac{\pi}{2}} \int_0^1 r^3 \cos^2 dr\, d\theta$.

1. Enter the "outer" limits: ['][←][π][÷][4][ENTER]['][←][π][÷][2][ENTER].
3. Enter the "inner" limits: [0][ENTER][1][ENTER].
4. Enter the integrand and "inner" variable of integration: ['][α][←][R] [y^x][3][×][COS][α][→][F][▶][y^x][2][ENTER]['][α][←][R][ENTER].
5. Evaluate the "inner" integral: [→][∫][EVAL].

   Result: '.25*COS(θ)^2'
6. Enter the "outer" variable of integration and, in Rad mode, evaluate the outer integral to find the volume: ['][α][→][F][ENTER][→][∫] ([←] [RAD], if necessary) [←][→NUM]. Result: 3.56747704246E-2.

However, if the function is given in rectangular form, but the region of integration R is given in polar form, then you'll first need to convert the rectangular function to a polar function using the following variable substitutions:

$$x = r \cos \theta \qquad y = r \sin \theta \qquad dx\, dy = r\, dr\, d\theta$$

(The r term in the latter equation is the Jacobian determinant scale factor for the transformation to cylindrical coordinates.)

Example: Find $\iint_R \left(1+x^2+y^2\right)^{\frac{3}{2}}\, dy\,dx$, with R the interior of the unit circle.

1. Analyze. In polar coordinates, the interior of the unit circle is described by $0 \le r \le 1$ and $0 \le \theta \le 2\pi$. After making the substitutions and simplifying (e.g. $x^2 + y^2 = r^2$), you have $\int_0^{2\pi}\int_0^1 \left(1+r^2\right)^{\frac{3}{2}} r\,dr\,d\theta$.
2. Enter the "outer" limits: [0] [ENTER] ['] [2] [×] [←] [π] [ENTER].
3. Enter the "inner" limits: [0] [ENTER] [1] [ENTER].
4. Enter the integrand and inner integration variable: ['] [←] [()] [1] [+] [α] [←] [R] [y^x] [2] [▶] [y^x] [1] [•] [5] [×] [α] [←] [R] [ENTER] ['] [α] [←] [R] [ENTER].
5. Evaluate the "inner" integral: [→] [∫].

   Result: `'∫(0, 1, (1+r^2)^1.5*r, r'`

   Note that the inner integral failed to evaluate in its current form.
6. Choose option b (see page 260) and use `CHVAR` to convert the integral, with $u = 1 + r^2$: ['] [α] [←] [U] [←] [=] [1] [+] [α] [←] [R] [y^x] [2] [ENTER] [α] [α] [C] [H] [V] [A] [R] [ENTER]. Result: `'∫(1, 2, .5*u^1.5, u)'`
7. Evaluate the revised inner integral and substitute the original expression for u, if needed: [EVAL] [EVAL]. Result: `.9313708499`

   Note that the evaluated inner integral contains no u (or other variable for that matter) and thus requires no resubstitution for u.
8. Enter the "outer" variable of integration and evaluate the outer integral to find the volume: ['] [α] [→] [F] [ENTER] [→] [∫] [←] [→NUM].

   Result: `5.85197563963`

Triple Integrals of Scalar Functions

The triple integral logically extends the concepts of single and double integration. The general form of a triple integral is $\iiint_S f(x,y,z)\,dV$, where dV is a volume element whose form depends on the order of the three integrals.

You can visualize the triple integral as computing an aggregate property (density, mass, temperature, etc.) of a solid, where that property *varies* locally throughout. Or, the triple integral is another way to compute a volume. Because volume—a "property" of every arbitrarily tiny particle of a solid—is a constant, triple-integrating $f(x,y,z) = 1$ yields the volume of the region. That is, given a region R bounded by two surfaces $z_1 = f(x,y)$ and $z_2 = g(x,y)$, over an area A defined by $h_1(y) \le x \le h_2(y)$ and $c_1 \le y \le c_2$, the volume of R is $\iiint_R 1\,dV = \int_{c_1}^{c_2}\int_{h_1(y)}^{h_2(y)}\int_{z_2}^{z_1} 1\,dz\,dx\,dy$.

Example: Find the volume of the region bounded between $z = x^2 + y^2$ and $z = -2xy$ over the area defined by $0 \le x \le y^2$ and $0 \le y \le 1$.

1. Outermost limits; middle limits: [0][ENTER][1][ENTER]; [0][ENTER]['] [α][←][Y][y^x][2][ENTER].
2. Innermost limits ($x^2 + y^2 \ge -2xy$ since x and y are ≥ 0): ['][+/–][2][×] [α][←][X][×][α][←][Y][ENTER]['][α][←][X][y^x][2][+][α][←][Y][y^x][2][ENTER].
3. The integrand; the innermost integration variable; then integrate: [1] [ENTER]['][α][←][Z][ENTER][→][∫][EVAL]. Result: `'x^2+y^2+2*x*y'`
4. The middle integration variable; then integrate: ['][α][←][X][ENTER] [→][∫][EVAL]. Result: `'2*(y^2^2/2)*y+y^2*y^2+y^2^3/3'`
5. The outer integration variable; then integrate to get the volume: ['] [α][←][Y][ENTER][→][∫][←][→NUM]. Result: `.414285714286`

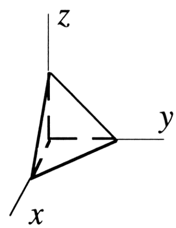
   Rationalizing this result (via `→Q` or `→Qπ`) yields `'29/70'`.

There are other ways to find the volume of an object, but a triple integral is the *only* way to find the total effect of a property that varies throughout the object.

Look, for example at density. If an object is a collection of arbitrarily small particles, and if each infinitesimal volume is the same, then any variation in its density is due to variation in its mass. And to find the mass of the total object, you need only sum the masses of its many infinitesimally particles—via triple integration.

Example: One vertex of a tetrahedron is at the origin (0,0,0). Its other vertices are at (0,0,1), (0,1,0), and (1,0,0). If the density p at each point (x,y,z) of the tetrahedron is $p(x,y,z) = xy$, what is the mass of the tetrahedron?

1. The tetrahedron is bounded between four planes—the xy-plane, the xz-plane, the yz-plane, and the plane $x + y + z = 1$:



The edges of the inclined face are lines: $y = 1–x$; $z = 1–y$; $x = 1–z$. Your integral can use the variables in any order (since the region is symmetrical), but for this case, let x be "outer," y "middle," and z "inner:" $0 \le x \le 1$ $\quad 0 \le y \le 1 - x$ $\quad 0 \le z \le 1 - x - y$

2. The outermost limits: [0] [ENTER] [1] [ENTER].
3. The middle limits: [0] [ENTER] ['] [1] [−] [α] [←] [X] [ENTER].
4. The innermost limits: [0] [ENTER] ['] [1] [−] [α] [←] [X] [−] [α] [←] [Y] [ENTER].
5. Enter the integrand and the innermost variable of integration, and evaluate the inner integral: ['] [α] [←] [X] [×] [α] [←] [Y] [ENTER] ['] [α] [←] [Z] [ENTER] [→] [∫] [EVAL]. Result: `'x*y*(1-x-y)'`
6. Enter the middle integration variable; evaluate the middle integral: ['] [α] [←] [Y] [ENTER] [→] [∫] [EVAL] [←] [SYMBOLIC] [COLCT] [EVAL] [EVAL].
   Result: `'-(x*((1-x)^3/3))-.5*(1-x)^2*x^2 +.5*(1-x)^2*x'`
7. Enter the outer integration variable; integrate to get the volume: ['] [α] [←] [X] [ENTER] [→] [∫] [←] [→NUM]. Result: `8.33333333336E-3`.
   Rationalizing the decimal result (with `→Q` or `→Qπ`) yields `'1/120'`.

Triple Integrals in Polar Coordinates

As with double integration, it may be convenient to use polar functions to describe the object or the area of integration. *Cylindrical* coordinates (r, θ, z) produce volume elements that are tiny cylindrical wedges of volume, $dV = r\,dr\,d\theta\,dz$.* To compute a triple integral using cylindrical coordinates, you must use this version of dV and express all limits and the integrand in cylindrical coordinates. Use these conversions, as needed: $x = r\cos\theta$ $\quad y = r\sin\theta$ $\quad z = z$ $\quad dx\,dy\,dz = r\,dr\,d\theta\,dz$.

Example: A cylindrical shell whose inner surface is $x^2 + y^2 = 1$, outer surface is $x^2 + y^2 = 4$ and which lies between the horizontal planes $z = 0$ and $z = 2$ is composed of a material whose density is $p(x,y,z) = x^2z + y^2z$. Find the total mass of the cylindrical shell.

1. In cylindrical coordinates, the inner surface is $r = 1$; the outer surface is $r = 2$ (because $x^2 + y^2 = r^2$). Similarly, the density function becomes $z(x^2 + y^2) = zr^2$. Thus, you must solve $\displaystyle\int_0^2\int_0^{2\pi}\int_1^2 zr^3\,dr\,d\theta\,dz$.
2. The limits—outer to inner: [0] [ENTER] [2] [ENTER]; [0] [ENTER] ['] [2] [×] [←] [π] [ENTER]; [1] [ENTER] [2] [ENTER].
3. The integrand and inner variable; integrate: ['] [α] [←] [Z] [×] [α] [←] [R] [y^x] [3] [ENTER] ['] [α] [←] [R] [ENTER] [→] [∫] [EVAL]. <u>Result</u>: 'z*4-z*.25'
4. The middle variable; then integrate: ['] [α] [→] [F] [ENTER] [→] [∫] [EVAL] [←] [SYMBOLIC] [COLCT]. <u>Result</u>: '7.5*z*π'
5. The outer variable; integrate to find the mass: ['] [α] [←] [Z] [ENTER] [→] [∫] [EVAL]. <u>Result</u>: '15*π'

Spherical coordinates (r, θ, ϕ) produce volume elements that are tiny spherical wedges, $dV = r^2\sin\phi\,dr\,d\theta\,d\phi$.** To compute a triple integral in cylindrical coordinates, you must use this version of dV and express all limits and the integrand in cylindrical coordinates. Use these conversions, as needed: $x = r\cos\theta\sin\phi$
$y = r\sin\theta\cos\phi$ $\quad z = r\cos\phi$ $\quad dx\,dy\,dz = r^2\sin\phi\,dr\,d\theta\,d\phi$

*The r term is the Jacobian determinant scale factor for the transformation to cylindrical coordinates.
**The $r^2\sin\phi$ term is the Jacobian determinant scale factor for the transformation to spherical coordinates.

Example: Find the mass of a solid ball of radius 1 if the density at each point d units from the center is $\dfrac{1}{1+d^2}$. Note that for a unit sphere, $0 \le r \le 1$, $0 \le \theta \le 2\pi$, and $0 \le \phi \le \pi$.

1. In spherical coordinates, the distance from the center is r; the density is $\dfrac{1}{1+r^2}$. Thus, you must solve $\displaystyle\int_0^{\pi}\int_0^{2\pi}\int_0^{1}\frac{r^2}{1+r^2}\sin\phi\, dr\, d\theta\, d\phi$.
2. Enter the outermost limits: [0] [ENTER] ['] [←] [π] [ENTER].
3. Enter the middle limits: [0] [ENTER] ['] [2] [×] [←] [π] [ENTER].
4. Enter the innermost limits: [0] [ENTER] [1] [ENTER].
5. Enter the integrand and the innermost variable of integration and evaluate the inner integral: ['] [α] [←] [R] [yˣ] [2] [ENTER] [ENTER] [1] [+] [÷] ['] [SIN] [α] [O] [α] [→] [9] [ENTER] [×] ['] [α] [←] [R] [ENTER] [→] [∫] [EVAL].

   Result: `'∫(0,1,r^2/(r^2+1)*SIN(Ø),r)'`
6. Because the inner integral failed to evaluate symbolically, try modifying it and re-evaluating. Note that `SIN(Ø)` is a constant in this integral and can be removed from the integrand (to be multiplied back after you evaluate). Then substituting $u = r^2 + 1$ (using `CHVAR`) converts the remaining integrand to an evaluable one. Make the changes and evaluate: [←] [EDIT] [▼] [◀] [◀] [DEL] [DEL] [DEL] [DEL] [DEL] [DEL] [DEL] [ENTER] ['] [α] [←] [U] [←] [=] [α] [←] [R] [yˣ] [2] [+] [1] [ENTER] [α] [α] [C] [H] [V] [A] [R] [ENTER] [EVAL]. Result: `'∫(1,2,.5*√(-1+u)/u,u)'`
7. The integral still won't evaluate. But since there are no variables in the integrand or the limits other than the variable of integration, you can use numeric evaluation (then restore the `SIN(Ø)` factor you removed: [←] [→NUM] ['] [SIN] [α] [O] [α] [→] [9] [ENTER] [×].

   Result: `'.21460183660З*SIN(Ø)'`
8. Enter the middle variable, then integrate: ['] [α] [→] [F] [ENTER] [→] [∫] [EVAL]. Result: `'.21460183660З*SIN(Ø)*(2*π)'`
9. The outer variable; integrate to get the mass: ['] [α] [O] [α] [→] [9] [ENTER] [→] [∫] [EVAL] [←] [SYMBOLIC] [COLCT]. Result: `'.858407346412*π'`

Using Triple Integrals to Find Averages

Another use for triple integrals is to compute the *average value* of a property that varies within a region. For example, the average density of an object of density $p = f(x,y,z)$ is: $\text{Average density} = \dfrac{\text{Total mass}}{\text{Total Volume}} = \dfrac{\iiint\limits_R p(x,y,z)\,dx\,dy\,dz}{\iiint\limits_R 1\,dx\,dy\,dz}$.

Example: Find the average density for an object described by $0 \le x \le y$, $0 \le y \le z$, and $0 \le z \le 1$ whose density function is $p(x,y,z) = xy^2z^3$.

1. Analyze the task. The problem reduces to $\dfrac{\int_0^1\int_0^z\int_0^y xy^2z^3\,dx\,dy\,dz}{\int_0^1\int_0^z\int_0^y 1\,dx\,dy\,dz}$.
2. Enter the iterated triple integral of the numerator; make a copy: ['] [→] [∫] [0] [←] [,] [1] [←] [,] [→] [∫] [0] [←] [,] [α] [←] [Z] [←] [,] [→] [∫] [0] [←] [,] [α] [←] [Y] [←] [,] [α] [←] [X] [×] [α] [←] [Y] [yˣ] [2] [×] [α] [←] [Z] [yˣ] [3] [←] [,] [α] [←] [X] [▶] [←] [,] [α] [←] [Y] [▶] [←] [,] [α] [←] [Z] [ENTER] [ENTER].
3. Use `MLTINT` to find the numerator integral: [α] [α] [M] [L] [T] [I] [N] [T] [ENTER]. <u>Result</u>: `1.11111111111E-2`
4. Swap the copy into level 1; edit the integrand to match the denominator, then evaluate the denominator triple integral using `MLTINT`: [SWAP] [←] [EDIT] [▼] [DEL] [DEL] [DEL] [DEL] [DEL] [DEL] [DEL] [DEL] [DEL] [1] [ENTER] [α] [α] [M] [L] [T] [I] [N] [T] [ENTER]. <u>Result</u>: `.166666666666`.
5. Divide and rationalize (using `→Qπ`) to finish the computation: [÷] [α] [α] [→] [→Q] [Q] [←] [C] [ENTER]. <u>Result</u>: `'1/15'`

The *centroid* of a object $(\bar{x},\bar{y},\bar{z})$ is the *average location* of all of its particles. If the object has a geometric center, then the centroid and the center are identical. The *center of mass* of an object $(\bar{x}_M,\bar{y}_M,\bar{z}_M)$ is the "balance point" of the object.

That is, if you were to support an object on a fulcrum placed exactly under the center of mass, the object would balance perfectly. If the object has constant density, then the center of mass is also the centroid; if the density varies, the center of mass will be distinct from the centroid. Both the centroid and center of mass are averages, so it isn't surprising that they are computed via triple integrals:

$$\text{Centroid} = \frac{\text{Total Position}}{\text{Total Volume}} \text{ or } (\bar{x},\bar{y},\bar{z}) = \left(\frac{\iiint\limits_R x\,dV}{\iiint\limits_R 1\,dV}, \frac{\iiint\limits_R y\,dV}{\iiint\limits_R 1\,dV}, \frac{\iiint\limits_R z\,dV}{\iiint\limits_R 1\,dV} \right)$$

$$\text{Center of mass} = \frac{\text{Total Position Mass}}{\text{Total Mass}} \text{ or}$$

$$(\bar{x}_M,\bar{y}_M,\bar{z}_M) = \left(\frac{\iiint\limits_R xp(x,y,z)\,dV}{\iiint\limits_R p(x,y,z)\,dV}, \frac{\iiint\limits_R yp(x,y,z)\,dV}{\iiint\limits_R p(x,y,z)\,dV}, \frac{\iiint\limits_R zp(x,y,z)\,dV}{\iiint\limits_R p(x,y,z)\,dV} \right)$$

Example: Find the centroid and center of mass of a solid bounded by the cylindrical surface $x^2 + z = 4$, the plane $x + z = 2$, and by the planes $y = 0$ and $y = 3$, and whose density varies according to $p(x,y,z) = xyz$.

1. The intervals for y and z are easy: $0 \le y \le 3$ and $2 - x \le z \le 4 - x^2$. To find the x-interval, find the points of intersection of the surface and the plane by solving a system of two equations: $2 - x = 4 - x^2$ or $x^2 - x - 2 = 0$. Thus, for the integral, $-1 \le x \le 2$.
2. Enter the denominator integral for the centroid; make three copies: ['] [→] [∫] [0] [←] [,] [3] [←] [,] [→] [∫] [1] [+/−] [←] [,] [2] [←] [,] [→] [∫] [2] [−] [α] [←] [X] [←] [,] [4] [−] [α] [←] [X] [y^x] [2] [←] [,] [1] [←] [,] [α] [←] [Z] [▶] [←] [,] [α] [←] [X] [▶] [←] [,] [α] [←] [Y] [ENTER] [ENTER] [ENTER] [ENTER].
3. Use MLTINT to evaluate the denominator triple integral: [α] [α] [M] [L] [T] [I] [N] [T] [ENTER]. Result: 13.5

4. Make two additional copies of the result, combine the three copies into a list and roll the list up to level 4 of the stack for later use: [ENTER] [ENTER] [3] [PRG] **LIST** **→LIST** [4] [↰] [STACK] **ROLLD**.
5. Edit the integrand of each of the copies of the triple integral so that each matches one of the numerator triple integrals in the centroid, and then combine the three numerators into a list: [↰] [EDIT] [▼] [▶] [▶] [▶] [▶] [▶] [▶] [▶] [DEL] [α] [↰] [X] [ENTER] **ROT** [↰] [EDIT] [▼] [▶] [▶] [▶] [▶] [▶] [▶] [▶] [DEL] [α] [↰] [Y] [ENTER] **ROT** [↰] [EDIT] [▼] [▶] [▶] [▶] [▶] [▶] [▶] [▶] [DEL] [α] [↰] [Z] [ENTER] [3] [PRG] **LIST** **→LIST**.
6. To find the centroid, use a short routine and `DOLIST`: [2] [ENTER] [↰] [« »] [α] [α] [M] [L] [T] [I] [N] [T] [α] [↰] [SWAP] [÷] [α] [α] [↱] [→] [Q] [↰] [C] [ENTER] **PROC** **DOLIS**. <u>Result</u>: `{ '1/2' '3/2' '12/5' }`

   So the centroid of the given object is the point $\left(\frac{1}{2}, \frac{3}{2}, \frac{12}{5}\right)$.
7. Enter the denominator integral for the center of mass; make three copies: ['] [↱] [∫] [0] [↰] [,] [3] [↰] [,] [↱] [∫] [1] [+/−] [↰] [,] [2] [↰] [,] [↱] [∫] [2] [−] [α] [↰] [X] [↰] [,] [4] [−] [α] [↰] [X] [y^x] [2] [↰] [,] [α] [↰] [X] [×] [α] [↰] [Y] [×] [α] [↰] [Z] [↰] [,] [α] [↰] [Z] [▶] [↰] [,] [α] [↰] [X] [▶] [↰] [,] [α] [↰] [Y] [ENTER] [ENTER] [ENTER] [ENTER].
8. Use `MLTINT` to evaluate the denominator triple integral: [α] [α] [M] [L] [T] [I] [N] [T] [ENTER]. <u>Result</u>: `15.1875000004`
9. Make two additional copies of the result, combine the three copies into a list and roll the list up to level 4 of the stack for later use: [ENTER] [ENTER] [3] [PRG] **LIST** **→LIST** [4] [↰] [STACK] **ROLLD**.
10. Edit the integrand of each of the copies of the triple integral so that each matches one of the numerator triple integrals in the center of mass, and then combine the three numerators into a list: [↰] [EDIT] [▼] [▶] [▶] [▶] [▶] [▶] [▶] [▶] [▶] [y^x] [2] [ENTER] **ROT** [↰] [EDIT] [▼] [▶] [▶] [▶] [▶] [▶] [▶] [▶] [▶] [▶] [▶] [y^x] [2] [ENTER] **ROT** [↰] [EDIT] [▼] [↱] [▶] [◀] [◀] [◀] [◀] [◀] [◀] [◀] [y^x] [2] [ENTER] [3] [PRG] **LIST** **→LIST**.
11. Find the center of mass using a short routine and `DOLIST`: [2] [ENTER] [↰] [« »] [α] [α] [M] [L] [T] [I] [N] [T] [α] [↰] [SWAP] [÷] [α] [α] [↱] [→] [Q] [↰] [C] [ENTER] **PROC** **DOLIS**. <u>Result</u>: `{ '52/35' '2/1' '9/5' }`

    So the center of mass of the given object is the point $\left(\frac{52}{35}, 2, \frac{9}{5}\right)$.

Path Integrals

So far in this course, the *interval of integration* has been measured along a straight line, usually one of the coordinate axes. But that's not always the case. The interval of integration is the *path* along which an object moves as it is subjected to a scalar or vector function. When the path is a straight line, the integral is a definite integral, measured in units of one the variables. When this path is curved, the integral is a *path* integral (or, misleadingly, a *line* integral) and the interval is measured in units of *arclength.* In that case, you must convert linear interval elements, such as *dt*, *dx*, *dy*, or *dz*, to arclength interval elements, usually notated as *ds*.

If you use a vector function of a single linear variable to describe a continuous curved path, $\mathbf{r}(t) = \{\ x(t)\quad y(t)\quad z(t)\ \}$, then the arclengths between two points on the curve is $s = \int_a^b \left\|\mathbf{r}'(t)\right\| dt$. The arclength formula can be expressed as a function, $s(t)$, that gives the arclength along $\mathbf{r}$ between a fixed start point $u = a$ and an arbitrary endpoint $u = t$: $s(t) = \int_a^t \left\|\mathbf{r}'(u)\right\| du$. Then, taking the derivative of the arclength function with respect to t and solving for the arclength interval element, *ds*, you get $ds = \left\|r'(t)\right\| dt$,which gives the conversion needed to define the path integral for scalar and vector functions: The path integral for a scalar function f along a parametrized curve, $\mathbf{r}(t)$, is $\oint_C f\, ds = \int_a^b f(\mathbf{r}(t))\left\|\mathbf{r}'(t)\right\| dt$; the path integral for a vector function $\mathbf{F}$ along a parametrized curve $\mathbf{r}(t)$ is $\oint_C \mathbf{F} \cdot ds = \int_a^b \mathbf{F}(\mathbf{r}(t)) \cdot \mathbf{r}'(t)\, dt$.*

The circle on the integral symbol and the small C indicate a path integral without specifying endpoints (and the interval element is *ds*). The C stands for "Curve," which allows you to speak generally of the path without necessarily having to specify a particular parametrization $\mathbf{r}$ of that path.

*You will sometimes see the path integral for a vector function expressed as $\oint_c P\, dx + Q\, dy + R\, dz$, where P, Q, and R are the component functions of the vector function.

Path (a.k.a. line) integrals are useful whenever the function in question acts on an object as it traces out a non-linear path, or when the physical quantity being measured depends on the length of a non-linear path. Some common examples:

- The mass of a wire with a density function f, traversing a curve C: $\oint_C f\,ds$
- The length of a wire traversing a curve C: $\oint_C 1\,ds$
- The work performed on an object over a curve C by a force $\mathbf{F}$: $\oint_C \mathbf{F}\cdot ds$
- The circulation along a curve C of a fluid moving according to $\mathbf{F}$: $\oint_C \mathbf{F}\cdot ds$

Compute some path integrals:

Example: Find the mass of a wire whose density function is $d(x,y,z) = x + y + z$ that traverses the path, $\mathbf{r}(t) = \{\ t\ \ t^2\ \ t^3\ \}$ where $0 \le t \le 1$.

1. Enter the limits of the parameter variable: [0] [ENTER] [1] [ENTER].
2. Enter the function: ['] [α] [←] [X] [+] [α] [←] [Y] [+] [α] [←] [Z] [ENTER].
3. Enter the path and make an extra copy: [←] [{ }] [α] [←] [T] ['] [α] [←] [T] [y^x] [2] [▶] ['] [α] [←] [T] [y^x] [3] [ENTER] [ENTER].
4. Enter the list of function variables and store the path components in them: [←] [{ }] [α] [←] [X] [SPC] [α] [←] [Y] [SPC] [α] [←] [Z] [ENTER] [STO].
5. Find the derivative of the path: ['] [α] [←] [T] [ENTER] [ENTER] [←] [PURG] [α] [α] [V] [D] [E] [R] [ENTER].
6. Assemble the integrand for the path integral: [α] [α] [V] [A] [B] [S] [ENTER] [SWAP] [EVAL] [×].
7. Enter the variable of integration, set the display mode to `5 FIX`, and integrate: ['] [α] [←] [T] [ENTER] [5] [α] [α] [F] [I] [X] [ENTER] [→] [∫] [←] [→NUM].

<u>Result</u>: `2.71478`

Example: Find the work done on an object by a force $\mathbf{F}(x,y,z) = \{\ y\ \ 2x\ \ y\ \}$ as it moves along the path, $\mathbf{r}(t) = \{\ t\ \ t^2\ \ t^3\ \}$ where $0 \le t \le 1$.

1. Enter the limits of the parameter variable: [0] [ENTER] [1] [ENTER].
2. Enter the function: [←] [{ }] [α] [←] [Y] ['] [2] [×] [α] [←] [X] [▶] [α] [←] [Y] [ENTER].
3. Enter the path and make an extra copy: [←] [{ }] [α] [←] [T] ['] [α] [←] [T] [y^x] [2] [▶] ['] [α] [←] [T] [y^x] [3] [ENTER] [ENTER].
4. Enter the list of function variables and store the path components in them: [←] [{ }] [α] [←] [X] [SPC] [α] [←] [Y] [SPC] [α] [←] [Z] [ENTER] [STO].
5. Compute the derivative of the path: ['] [α] [←] [T] [ENTER] [ENTER] [←] [PURG] [α] [α] [V] [D] [E] [R] [ENTER].
6. Assemble the integrand for the path integral: [SWAP] [1] [« »] [EVAL] [▼] [PRG] [LIST] [PROC] [DOLIS] [ENTER] [SWAP] [α] [α] [S] [D] [O] [T] [ENTER] [←] [SYMBOLIC] [COLCT].
7. Enter the integration variable and then integrate: ['] [α] [←] [T] [ENTER] [→] [∫] [EVAL].

   Result: 2.2667, which converts to '34/15' via →Q.

The program PATHINT (page 306) allows you to compute a path integral with a minimum of "calculator overhead." It takes the function (either scalar or vector) from level 5, the parametrized curve (as a symbolic vector) from level 4, the list of independent variables in the function from level 3, the parameter variable for the curve from level 2, and a list containing the starting and ending values of the parameter variable from level 1. PATHINT returns the computed path integral to level 1 at the accuracy level set by the current display mode.

Example: Use PATHINT to find $\oint_C f\, ds$, where $f(x,y,z) = x + \cos^2 z$, and the curve C is parametrized by $\mathbf{r}(t) = \{\ \sin t\ \ \cos t\ \ t\ \}$ where $0 \le t \le 2\pi$.

1. First, enter the function itself: ['] [α] [←] [X] [+] [COS] [α] [←] [Z] [▶] [y^x] [2] [ENTER].
2. Next, enter the parametrized curve: [←] [{ }] ['] [SIN] [α] [←] [T] [▶] [▶] ['] [COS] [α] [←] [T] [▶] [▶] ['] [α] [←] [T] [ENTER].

3. Enter the list of function variables: [↰][{ }][α][↰][X][SPC][α][↰][Y][SPC][α][↰][Z][ENTER].
4. Enter the parameter variable: ['][α][↰][T][ENTER].
5. Enter the range of the parameter variable as a list: [↰][{ }][0][SPC]['][2][×][↰][π][ENTER].
6. Set the display mode to STD and compute the path integral: [α][α][S][T][D][ENTER], then [α][α][P][A][T][H][I][N][T][ENTER] or [VAR] (then [NXT] or [↰][PREV] as needed) [PATHI].

<u>Result</u>: 4.44288293815

This matches the analytic answer, $\pi\sqrt{2}$, to full precision.

Example: Find $\oint_C \mathbf{F} \cdot ds$ where $\mathbf{F}(x,y,z) = \{\ y^2z^2 \quad \cos(x^2 + y^2) \quad \ln(xyz)\ \}$ and the curve C is parameterized by $\mathbf{r}(t) = \{\ t^3/3 \quad t^2 \quad 4t\ \}$ where $1 \le t \le 2$.

1. Enter the vector function: [↰][{ }]['][α][↰][Y][y^x][2][×][α][↰][Z][y^x][2][▶]['][COS][α][↰][X][y^x][2][+][α][↰][Y][y^x][2][▶][▶]['][↱][LN][α][↰][X][×][α][↰][Y][×][α][↰][Z][ENTER].
2. Enter the parametrized curve: [↰][{ }]['][α][↰][T][y^x][3][÷][3][▶]['][α][↰][T][y^x][2][▶]['][4][×][α][↰][T][ENTER].
3. Enter the list of function variables: [↰][{ }][α][↰][X][SPC][α][↰][Y][SPC][α][↰][Z][ENTER].
4. Enter the parameter variable: ['][α][↰][T][ENTER].
5. Enter the range of the parameter variable as a list: [↰][{ }][1][SPC][2][ENTER].
6. Compute the path integral: [α][α][P][A][T][H][I][N][T][ENTER] or [VAR] ([NXT] or [↰][PREV]) [PATHI].

<u>Result</u>: 918.534069341

Potentials

The gradient of a scalar function, f, is a vector function, $\mathbf{d}_f$. But how about the reverse operation? Can you find a scalar function that has $\mathbf{d}_f$ as its gradient?

If such a scalar function exists, it's called a *potential* of $\mathbf{d}_f$. Potentials are antiderivatives for vector functions. Just as scalar functions have an infinite number of anti-derivatives (as long as they have at least one), vector functions have an infinite number of potentials (provided that they have at least one). For a potential to exist, antiderivatives must exist for each of the vector function's component functions.

The program POTEN (page 314) takes the vector function from level 2 and the list of variables from level 1 and returns the scalar potential to level 1. POTEN uses INDEF to look for antiderivatives and thus has the same limited pattern-matching capabilities as the built-in commands. And you can enhance those abilities by substituting XINDEF for INDEF in the POTEN program and adding integration patterns to IPATS using ADDPAT, if you wish (see page 175 for details).

Example: Use POTEN to find a potential for $\mathbf{F}(x,y,z) = \{\ 3x^2y\ \ x^3 + y^3\ \ 2z\ \}$.

1. Enter the vector function: [←][{ }]['][3][×][α][←][X][yˣ][2][×][α][←][Y] [▶]['][α][←][X][yˣ][3][+][α][←][Y][yˣ][3][▶]['][2][×][α][←][Z][ENTER].
2. Enter the variables list: [←][{ }][α][←][X][SPC][α][←][Y][SPC][α][←][Z] [ENTER].
3. Find a potential: [α][α][P][O][T][E][N][ENTER] or [VAR] ([NXT] or [←][PREV] as needed) [POTE].

   Result: `'x^3*y+.25*y^4+z^2'`

A vector function is said to be *conservative* if it has at least one potential. But being a conservative vector function also means that its path integral equals zero *for every closed curve C.* This means that the path integral of a conservative vector function depends upon only the two endpoints, not on the particular path traveled. Furthermore, if you start and end at the same point, the path integral of a conservative vector function always equals zero—no matter how long or intricate the circular path.

So, if you can find a potential for a vector function **F**, you know it's a conservative function. And you can show that a vector function **F** has no potentials if you can show that it isn't conservative. Is there a quick and easy way to prove that a vector function **F** isn't conservative?

Yes. If the curl of **F** isn't the zero vector, then **F** isn't conservative. However, if the curl of **F** is the zero vector, it neither guarantees that **F** is conservative nor that it has a potential (although it's a good sign that it's worth looking for one).

Example: Use the Curl test to determine if $\mathbf{F}(x,y,z) = \{\ 3x^2y\ \ x^3 + y^3\ \ 2z\ \}$ could be conservative.

1. Enter the vector function: [←][{ }]['][3][×][α][←][X][y^x][2][×][α][←][Y][▶]['][α][←][X][y^x][3][+][α][←][Y][y^x][3][▶]['][2][×][α][←][Z][ENTER].
2. Enter the variables list: [←][{ }][α][←][X][SPC][α][←][Y][SPC][α][←][Z][ENTER].
3. Compute the curl: [α][α][C][U][R][L] [ENTER].

   Result: { 0 0 0 }

   The zero vector result indicates that **F** *might* be conservative; the test doesn't prove that it is. It merely *suggests* what you already know for a fact from the previous example—that **F** is indeed conservative because it has a potential.

Surface Integrals

A path integral is a single integration along a parametrized curve (a single-variable vector function). Similarly, a *surface integral* is a double integration over a parametrized surface (a multi-variable vector function).

Furthermore, just as path integrals require the transformation of straight-line, variable-axis interval elements to curved arclength interval elements (ds), so do surface integrals require the transformation of *two-dimensional* area elements to *three-dimensional* surface area elements (dS).

When a flat rectangular (du by dv) element is parametrized into a surface area element, it's best approximated as a parallelogram whose sides are $du\|\mathbf{T}_u\|$ and $dv\|\mathbf{T}_v\|$, where $\|\mathbf{T}_u\|$ and $\|\mathbf{T}_v\|$ are the lengths of the two tangent vectors for the element. The area of the parallelogram with sides of lengths a and b is $ab \sin \theta$. Now, the *length* of the fundamental vector product is $\|\mathbf{T}_u \times \mathbf{T}_v\| = \|\mathbf{T}_u\|\|\mathbf{T}_v\|\sin\theta$. Put all together, this means that the surface area element (dS) can be transformed from its rectangular counterpart ($du\,dv$) by $dS = \|\mathbf{T}_u\|\|\mathbf{T}_v\|\sin\theta\,du\,dv = \|\mathbf{T}_u \times \mathbf{T}_v\|\,du\,dv$.

Thus, the surface area of a parametrized surface (analogous to the arclength of a parametrized curve) is $\iint_S dS = \iint_R \|T_u \times T_v\|\,du\,dv = \int_{v_0}^{v_1}\int_{u_o}^{u_1} \|T_u \times T_v\|\,du\,dv$.

Finally, the surface integrals for scalar and vector functions are defined as:

Scalar: $\iint_S f\,dS = \iint_R f(\mathbf{r}(u,v))\|\mathbf{T}_u \times \mathbf{T}_v\|\,du\,dv$, where f is the scalar function and $\mathbf{r}(u,v)$ is the parametrized surface.

Vector: $\iint_S \mathbf{F}\cdot d\mathbf{S} = \iint_R \mathbf{F}(\mathbf{r}(u,v))\cdot(\mathbf{T}_u \times \mathbf{T}_v)\,du\,dv$, where $\mathbf{F}$ is the vector function and $\mathbf{r}(u,v)$ is the parametrized surface.

Some common applications for surface integrals are:

- The area of a surface S: $\iint_S 1\, dS$
- The mass of a surface S whose density is $f(x,y,z)$: $\iint_S f\, dS$
- The flux of a fluid of velocity $\mathbf{F}(x,y,z)$ across a surface S: $\iint_S \mathbf{F} \cdot d\mathbf{S}$

Computing a surface integral requires the following steps:

1. Determine the parametrization of the surface and the ranges for its two variables.
2. Compute the fundamental vector product (a vector function) of the parametrized surface.
3. *For a scalar function*: Compute $f(\mathbf{r}(u,v))$ and multiply it by the absolute value of the fundamental vector product.

   For a vector function: Compute $\mathbf{F}(\mathbf{r}(u,v))$ and find the dot product of it and the fundamental vector product.
4. Compute the double integral using the result of step 3 as the integrand and the ranges of the parameter variables as the appropriate limits.

The program SRFINT (page 330) handles the management of the surface integral computations. It takes the function (either scalar or vector) from level 5, the parametrized surface (as a symbolic vector) from level 4, the list of function variables from level 3, and two 3-element lists on levels 2 and 1, declaring the names and ranges of the parameter variables. Each of these last two lists begins with the name of the parameter variable followed by the start and end points of its range.

SRFINT then sets up the correct double integral and attempts to evaluate the inner integral symbolically. If it fails to do so, it halts and displays the inner integral, to allow you to manipulate or resolve it by hand before pressing [⬅] [CONT] to resume the computation. This option to pause exists because SRFINT can take awhile (sometimes upwards of 20 minutes) to compute the surface integral— a double integral—if it can't symbolically evaluate the inner integral and has to resort to numerical double integration at reduced precision.

Example: Find the total mass of a material surface whose density function is $f(x,y,z) = xyz$ and which is parametrized by $\mathbf{r}(u,v) = \{\ u\ \ v\ \ uv\ \}$ where $0 \le u \le 1$ and $0 \le v \le 1$.

1. Enter the function: ['] [α] [←] [X] [×] [α] [←] [Y] [×] [α] [←] [Z] [ENTER].
2. Enter the parametrized surface: [←] [{ }] ['] [α] [←] [U] [▶] ['] [α] [←] [V] [▶] ['] [α] [←] [U] [×] [α] [←] [V] [ENTER].
3. Enter the list of function variables: [←] [{ }] [α] [←] [X] [SPC] [α] [←] [Y] [SPC] [α] [←] [Z] [ENTER] .
4. Enter a list containing the first parameter and its range: [←] [{ }] [α] [←] [U] [SPC] [0] [SPC] [1] [ENTER].
5. Enter a list with the second parameter and its range: [←] [{ }] [α] [←] [V] [SPC] [0] [SPC] [1] [ENTER].
6. Compute the surface integral: [α] [α] [S] [R] [F] [I] [N] [T] [ENTER] or [VAR] (then [NXT] or [←] [PREV] as needed) SRFIN.

   Result: `'∫(0,1,√(1+(-u)^2+(-v)^2)*u^2*v^2,u)'`
7. Because the integral doesn't look too complicated (and you're feeling lucky), continue the computation without making any changes: [←] [CONT].

   Result (to 4 places): `0.1642` (after about three minutes)

Example: If a fluid's velocity is given by $\mathbf{F}(x,y,z) = \left\{ -\dfrac{\sqrt{2}}{2} \quad 0 \quad -\dfrac{\sqrt{2}}{2} \right\}$, find the fluid's outward flux across the parametrized surface described by $\mathbf{r}(u,v) = \{\ u \cos v \quad u \sin v \quad v\ \}$, where $0 \le u \le 1$ and $0 \le v \le \pi/2$.

1. Enter the vector function: [←] [{ }] ['] [+/−] [√x] [2] [÷] [2] [▶] [0] ['] [+/−] [√x] [2] [÷] [2] [ENTER].
2. Enter the parametrized surface: [←] [{ }] ['] [α] [←] [U] [×] [COS] [α] [←] [V] [▶] [▶] ['] [α] [←] [U] [×] [SIN] [α] [←] [V] [▶] [▶] ['] [α] [←] [V] [ENTER].

3. Enter the list of function variables: [←][{ }][α][←][X] [SPC][α][←][Y] [SPC][α][←][Z][ENTER].
4. Enter a list containing the first parameter and its range: [←][{ }][α][←] [U][SPC][0][SPC][1][ENTER].
5. Enter a list with the second parameter and its range: [←][{ }][α][←][V] [SPC][0][SPC]['][←][π][÷][2][ENTER].
6. Compute the surface integral: [α][α][S][R][F][I][N][T][ENTER] or [VAR] (then [NXT] or [←][PREV] as needed) SRFIN.

Result: -1.26246714846

Whenever the ranges for the parameter variables don't describe a rectangular region, you may need to convert the parametrization to polar coordinates—just as you did when plotting parametrized surfaces back on page 238.

Example: Find the outward flux of $\mathbf{F}(x,y,z) = \{\ x\ \ y\ \ z\ \}$ over the upper unit hemisphere, $\mathbf{S}(u,v) = \{\ u\ \ \ v\ \ \sqrt{1-u^2-v^2}\ \}$, where $0 \leq u^2 + v^2 \leq 1$.

1. Analyze the task. Note that the integration region is not rectangular, but circular, so that transforming the problem to polar coordinates is in order. So letting $u = r \cos \theta$ and $v = r \sin \theta$ yields this parametrization: $\mathbf{S}(r,\theta) = \{\ r\cos\theta\ \ r\sin\theta\ \ \sqrt{1-r^2}\ \}$ where $0 \leq r \leq 1$ and $0 \leq \theta \leq 2\pi$.
2. Enter the vector function: [←][{ }][α][←][X][SPC][α][←][Y][SPC][α][←][Z] [ENTER].
3. Enter the parametrized surface: [←][{ }]['][α][←][R][×][COS][α][→][F] [▶][▶]['][α][←][R][×][SIN][α][→][F][▶][▶]['][√x][←][()][1][−][α][←][R][y^x] [2][ENTER].
4. Enter the list of function variables: [←][{ }][α][←][X] [SPC][α][←][Y] [SPC][α][←][Z][ENTER].
5. Enter a list containing the first parameter and its range: [←][{ }][α][←] [R][SPC][0][SPC][1][ENTER].
6. Enter a list with the second parameter and its range: [←][{ }][α][→][F] [SPC][0][SPC]['][2][×][←][π][ENTER].

7. Compute the surface integral: [α][α][S][R][F][I][N][T][ENTER] or [VAR] (then [NXT] or [←][PREV] as needed) **SRFIN**.

   Result: `'∫(0,1),∫(1-r^2)*SIN(θ)^2*r+∫(1-r^2)*COS(θ)^2*r+SIN(θ)^2/∫(1-r^2)*r^3+COS(θ)^2/∫(1-r^2)*r^3,r)'`

8. This integral looks too complicated for quick evaluation. Try simplifying the integrand:

$$
\begin{aligned}
& r\sin^2\theta\sqrt{1-r^2} + r\cos^2\theta\sqrt{1-r^2} + \frac{r^3\sin^2\theta}{\sqrt{1-r^2}} + \frac{r^3\cos^2\theta}{\sqrt{1-r^2}} \\
&= \left(\sin^2\theta+\cos^2\theta\right)r\sqrt{1-r^2} + \left(\sin^2\theta+\cos^2\theta\right)\left(\frac{r^3}{\sqrt{1-r^2}}\right) \\
&= r\sqrt{1-r^2} + \frac{r^3}{\sqrt{1-r^2}} = \frac{r\left(1-r^2\right)+r^3}{\sqrt{1-r^2}} \\
&= \frac{r}{\sqrt{1-r^2}}
\end{aligned}
$$

   Replace the complicated integrand with its simpler version: [←][EDIT] [▶][▶][▶][▶][▶][▶][▶] **DEL→** **DEL→** **DEL→** **DEL→** **DEL→** [α][←][R] [÷][√x][←][()][1][−][α][←][R][y^x][2][▶][←][,][α][←][R][←][()][◆][ENTER].

9. Now, although you've got a simpler integrand, it's one that appears well-suited for a u-substitution using $u=\sqrt{1-r^2}$. Enter the transformation equation and execute `CHVAR`: [']['α][←][U][←][=][√x][←][()][1][−][α][←][R][y^x][2][ENTER][α][α][C][H][V][A][R][ENTER].

   Result: `'∫(1,0,-1,u)'`

10. Ahhh—much better! Press [←][CONT] to finish the computation.

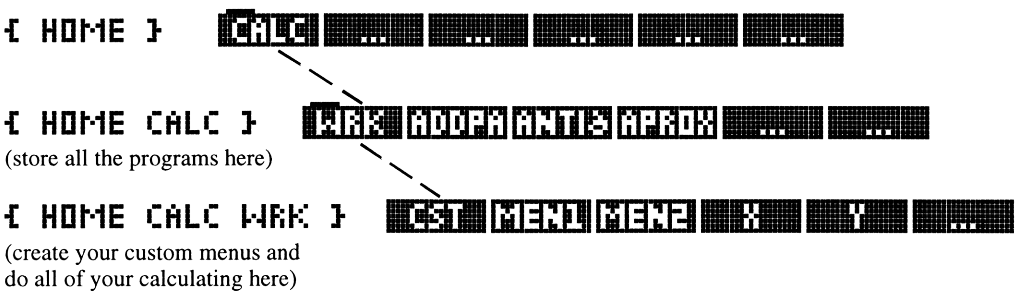
   Result: `6.28318530718` $\approx 2\pi$.

P. Program Listings

Before You Key In or Use These Programs

This Appendix contains a listing of all of the programs referred to throughout this book, sorted alphabetically by name (numerals after letters and special symbols ignored), with text page references noted opposite the name. To use a program by invoking its name, you must have it properly stored—in that name—within the current directory path. (*Note:* If you have an HP 48G, you won't be able to fit all of these programs into the 32K storage at once; you'll need to pick and choose.)

As with all HP 48 variables, you must be careful to avoid name conflicts with other variables in the current directory path. One suggestion: Put the programs into a subdirectory, then create a work space below that, with custom menus to help you organize and access the programs (for more about custom menus, see your user's manual or Grapevine Publications' *Easy Course in Using and Programming the HP 48G/GX*). This lets you work efficiently without corrupting your programs:



If you have a bit of programming aptitude, the programs can be modified to suit your tastes and/or needs. Most of them have not been rigorously groomed for error-trapping, speed, or memory efficiency; they are designed simply to work well with the examples in this course and with related work. Also, you may wish to modify the input or output of the programs. For example, geometric points may be expressed as either complex numbers or as two-element vectors, depending on the context in which you're working.

Whether you use these programs as is or otherwise, above all you should *practice* using them *before* needing them in an important situation. You must understand how they work, how fast are they, how to interpret their outputs, and the nature of their limitations (special cases of functions or flag settings).

Of course, each program is designed to work flawlessly, but bugs (and typos) are, unfortunately, facts of life with software and other creative works. If you have a problem with a program, you may contact the publisher, <u>but first, check again</u>:

- *Have you correctly entered the program(s)?* Some items to check:

  The program size (bytes) and checksum <u>must</u> match those shown. For example, the program ADDPAT, shown opposite, must have exactly 1036 bytes, with a checksum of #B259. To calculate these test numbers, first enter and name (i.e. store) the program. Then put its *name* (within ' ' marks) onto the stack and press ⬅ MEMORY BYTES.*

  If your byte-count/checksum results are different than those prescribed, you have a typo somewhere in your program. Common errors include:

  - Using uppercase vs. lowercase letters (yes, this is significant);
  - Miskeying special characters (use the ➡ CHARS tool);
  - STO vs. STO, 1 vs. 1, O vs. 0, or { } vs. ( ) vs. [ ]. Be careful!
  - Using ' ' vs. ". Quotes (") are on the ➡ − key—don't use ' '.
  - Putting spaces (or carriage returns) where they should not be. Space characters within " " are significant—count 'em if necessary (the uniform spacing of the program font makes this easy); all other indents, line breaks, etc., represent *single spaces*. These program listings are shown with indents and line breaks for your eyes only; the calculator does not use them. To it, a program is simply a series of objects, separated by single spaces, all on one long line; even the indents and line breaks in the HP 48 display when you edit are just for your benefit. So ignore indents, and where you see line breaks, just treat those as single spaces.
  - Some programs use ("call") other programs; **the called programs must also be properly keyed in and named.** Such instances appear here in ***Boldface Italics***. For example, the program ADDPAT, shown opposite, calls FUNOF?, LININ? and →RPN, so you must also key in those three programs before ADDPAT will run.

- *Are you correctly using the program?* Double-check the types and order of your inputs and the types and ranges of your graph settings. Note that each program listing shows the required order and types of inputs (if any)

*All checksums are binary integers. Those given in this book are all in HEX notation with a 64-bit wordsize. It is very convenient, therefore, to adjust your machine to that setting: Press MTH BASE HEX NXT 6 4 STWS.

1041 bytes #3625h

1:	====>	1:

```
«  -3 CF
   WHILE "ADD INTEGRAL PATTERN" { { "REPLACE:" "ENTER
      INTEGRAND TO BE MATCHED" 9 } { } { "WITH:" "ENTER
      REPLACEMENT PATTERN" 9 } { } { "EXPR. VAR:" "ENTER
      VAR. FOR LINEAR EXPRESSION" 6 } { "INT. VAR:" "ENTER
      VARIABLE OF INTEGRATION" 6 } { "CONSTANTS:" "ENTER LIST
      OF CONSTANTS" 5 } { } } { 2 2 } { NOVAL NOVAL x x
      NOVAL } DUP INFORM
   REPEAT OBJ→ DROP 5 ROLL 5 ROLL 2 →LIST
      →  u v c f
      «  IF c SIZE DUP
         THEN
            DUP 1 SWAP
            FOR k
               "'FUNOF?(QUOTE(&" k 1 + + "),↑↓)'" + OBJ→
               IF k 2 ≥ THEN OR END
            NEXT
            NOT SWAP
         END 'LININ?(QUOTE(&1),QUOTE(↑↓))' SWAP
         IF THEN AND END
         f 1
         «  →RPN 1
            «  IF DUP u SAME THEN DROP &1 END
               IF DUP v SAME THEN DROP ↑↓ END
               IF DUP c SWAP POS DUP
               THEN "&" SWAP 1 + + OBJ→ SWAP DROP
               ELSE DROP
               END
               IF DUP { ∂ } 1 GET SAME
               THEN DROP « "∂" SWAP + "." + SWAP + OBJ→ »
               END
            »
            DOLIST EVAL
         »
         DOLIST '∂↑↓(&1)' '∂↑↓.&1' STO 1 « '↑↓' SHOW »
         DOLIST
      »
      SWAP +
      IF IPATS DUP TYPE 6 ==
      THEN { } SWAP STO
      ELSE DROP
      END
      1 →LIST IPATS + 'IPATS' STO '∂↑↓.&1' PURGE
   END
»
```

ANTIS — Plot an Antiderivative Slopefield (172)

224 bytes #8552h

6: Symbolic expression for slope	6:
5: List of variables: {indep depnd}	5:
4: Plotting range: {begin end}	4:
3: Horizontal display range: {left right}	3:
2: Vertical display range: {low high}	2:
1: Initial conditions: (x_0,y_0) or list of such ordered pairs ====>	1:

```
« → f v prg xrg yrg init
  « ERASE v HEAD prg + INDEP xrg EVAL DUP2 XRNG XVOL v 2
    GET yrg + DEPND yrg EVAL YVOL 10 NUMX 8 NUMY
    SLOPEFIELD f STEQ DRAW f v prg init 131 EULPLT
  »
»
```

APROX — Compute a Taylor Series Approximation (46)

115 bytes #B089h

5: function being approximated	5:
4: independent variable	4:
3: order of Taylor series desired	3: Taylor series
2: point around which approx. is centered	2: Approximate value
1: point being approximated ====>	1: Estimated error

```
« 5 ROLLD 4 DUPN TYLRa
  → b f v o a p
  « b v STO p DUP →NUM f →NUM OVER - ABS v PURGE
  »
»
```

A→Q — Rationalize Elements of an Array (309)

211 bytes #5DE2h

1: array or symbolic array ====>	1: rationalized symbolic array

```
« RCLF -3 CF SWAP OBJ→ OBJ→
  IF 1 == THEN 1 SWAP END
  9 FIX
  → row col
  « 1 row
    FOR k
       1 col START 8 FIX →Q STD col ROLLD NEXT
       col →LIST col row k - * k + ROLLD
    NEXT
    IF row 1 > THEN row →LIST END
    SWAP STOF
  »
»
```

C1BC2 Compute the Angle Between Curves at Given Points (107)

325.5 bytes #FE66h

3: List of curves		3:
2: List of variables		2:
1: List of points (given as ordered pairs)	====>	1: List of angles

```
«  -3 CF
   → c v p
   «  c 1 « v IMPS » DOLIST
      → d
      «  1 2
         FOR j
            d j GET 1 p SIZE
            FOR k
               DUP p k GET C→R 2 →LIST v STO "'&" v 2
               GET + "." + v 1 GET + OBJ→ 0 ROOT SWAP
            NEXT
            DROP p SIZE →LIST
         NEXT
         2 « DUP2 - 1 4 ROLL 4 ROLL * + / ATAN ABS »
         DOLIST p 2 « "θ" SWAP + →TAG » DOLIST v PURGE
      »
   »
»
```

CDINT Determine the Convergence of an Improper Integral (163)

1053.5 bytes #A59Ch

4: Symbolic integral		4:
3: Name of limit variable		3:
2: Value of problematic limit		2:
1: -1/1 indicator for direction of limit	====>	1: string describing conclusion

```
«  4 PICK OBJ→ DROP2 RCLF { -3 -55 } CF MAXR 'ω' STO
   1 0 0 0 0 0
   → f v l s ↓a ↑b ig x fl d q n0 n1 dif v0
   «  x PURGE
      IF
         f v l 2 →LIST ↓MATCH DROP OBJ→ DROP2 DUP 3 ROLLD
         INDEF
      THEN
         XRCONV DUP 4 ROLL 4 PICK STO EVAL SWAP 4 ROLL 4
         ROLL STO EVAL - COLCT "Limit" →TAG
      ELSE
         4 DROPN '↓a' '↑b' ig x ∫ 'f' STO CLLCD 0
         IF s 1 - THEN ↓a ELSE ↑b END
         'v0' STO
         DO
            3 FIX 1 d * + DUP ALOG
            IF 1 TYPE 1 <
            THEN INV s * 1 +
            ELSE IF 1 →NUM SIGN -1 == THEN NEG END
            END
            v0
            IF s 1 - THEN '↓a' STO ELSE '↑b' STO END
```

*** *Be sure to read the instructions on pages 283-284 before keying in these programs.* ***

```
            →NUM v STO f EVAL →NUM v →NUM 'v0' STO DUP
            STO 'n1' STO n0 + DUP 1 DISP 'n0' STO
            IF
              n1 DUP SIGN dif SIGN == n0 1 < OR 'q' INCR
              3 ≤ OR
            THEN 'dif' STO
            ELSE DROP d 3 * - d .1 * 'd' STO
            END
         UNTIL
            n1 ABS .0000001 ≤ n0 ABS 100000 ≥ OR n0
            ABS .000001 ≤ OR q 10 > OR
         END
         DROP
         IF
            n0 ABS 100000 ≥ q 10 > n1 n0
            IFERR / THEN DROP MAXR * END
            ABS .001 > AND OR
         THEN "Diverges"
         ELSE "Converges" "~" + n0 3 FIX +
         END
      END
      v PURGE 'ω' PURGE fl STOF
   »
»
```

CHVAR Change Variables in an Integral (165)

444 bytes **# 876h**

2: Symbolic integral		2:	
1: Equation defining change of variables	====>	1: modified integral	

```
« DUP EQ→ 4 ROLL OBJ→ DROP2 RCLF -3 CF
  → y g low up f x flags
  « -1 SF x ISOL EQ→ SWAP DROP g x FDER
    → ginv dg
    « g x low 2 →LIST ↑MATCH DROP g x up 2 →LIST ↑MATCH
      DROP f dg / g y 2 →LIST ↑MATCH DROP EXCO x ginv 2
      →LIST ↑MATCH DROP y 4 →LIST ↑↓ APPLY {
      '↑↓(&1,&2,&3,&4)' '∫(&1,&2,&3,&4)' } ↓MATCH DROP
      flags STOF COLCT { '&1^&2^&3' '&1^(&2*&3)' } ↓MATCH
      DROP COLCT
    »
  »
»
```

CONTOUR Draw a Contour Plot of a Function (232)

313.5 bytes **#A35Ah**

4: Function of two variables		4:
3: List of independent variables		3:
2: List of range of contours		2:
1: Number of steps (negative if no erase first)	====>	1:

```
« DUP SIGN SWAP ABS VPAR 1 4 SUB EVAL YRNG 2 →LIST
  → f v z s n p
```

```
    « p EVAL XRNG z EVAL - ABS n /
      → d
      « 1 n FOR k f z 1 GET d k 1 - * + - NEXT
        n →LIST f z 2 GET - + v p f p HEAD v HEAD
        STO v 2 GET 0 ROOT
        1 n START DUP NEXT
        n 1 + →LIST 4 s * SOLVPLT v PURGE
      »
    »
»
```

CURL **Compute the Curl of a Vector Function** (254)

238 bytes #9B98h

2: Symbolic vector function		2:	
1: List of variables	====>	1: Curl (as symbolic vector)	

```
« -3 CF
  IF DUP SIZE 2 == THEN '↑↓' + SWAP 0 + SWAP END
  JACOB EVAL
  → p q r
  « 'r(2)-q(3)' EVAL 'p(3)-r(1)' EVAL 'q(1)-p(2)' EVAL 3
    →LIST
  »
»
```

CURV **Find the Curvature of a Parametrized Curve at a Point** (251)

129 bytes #F5CDh

3: Parametrized curve (symbolic vector)		3:
2: Parameter name		2: Symbolic curvature
1: Value of parameter at point	====>	1: Numeric Curvature

```
« -3 CF
  → f v x
  « f v VDER DUP DUP VABS / v VDER VABS SWAP VABS / x v
    STO →NUM v PURGE
  »
»
```

CVINT **Compute the Interval of Convergence for a Power Series** (48)

765 bytes # 81h

3: coefficients expression of power series		3:
2: index variable		2:
1: point about which series is computed	====>	1: "interval of convergence"

```
« -3 CF -22 SF DEPTH
  → f v a d
  « f ABS v XROOT
    → fp
```

```
«  1 26 FOR j j EXP →NUM v STO fp →NUM NEXT
   26 →LIST DUP
   IF
      { 3 4 } 1 « v STO fp →NUM » DOLIST OBJ→ DROP /
      1 >
   THEN « MIN »
   ELSE « MAX »
   END
   STREAM
   IF DUP
   THEN
      DUP2 POS ROT DROP DUP EXP DUP INV
      → t j d
      « DO
           j 0 t d LOG ABS 1 + CEIL - MAX EXP IP +
           DUP 'j' STO v STO fp →NUM DUP 1 DISP DUP
           ROT DUP2 OVER - ABS SWAP / 'd' STO
        UNTIL
           DUP2 == 3 ROLLD OVER 50 > 4 ROLLD DUP
           'p' STO - SIGN DUP s NEG == j 10 ≥ AND
           SWAP 's' STO OR KEY OR OR
        END
        DROP p
      »
   ELSE SWAP DROP
   END
   v PURGE { p s } PURGE
»
DEPTH 2 + d - DROPN
IF DUP 50 >
THEN DROP "x=" a +
ELSE
   IF DUP 1 RND 0 ==
   THEN DROP "All x"
   ELSE
      INV 2 RND DUP a + a ROT - "<" DUP 'x' + SWAP
      + + SWAP +
   END
END
   »
»
```

D2TST **Perform the Second-Order Derivative Test** (248)

318.5 bytes #A6FFh

3: Scalar multivariable function		3:	
2: List of variables		2:	
1: List of points (each given as ordered pair)	====>	1:	List of labeled points

```
«  → f v p
   « f 1 →LIST v JACOB EVAL v JACOB 1 « COLCT » DOLIST DUP
     SDET
     → m d
     « p 1
       «  C→R 2 →LIST v STO
          IF d EVAL DUP
          THEN
             IF 0 >
```

```
            THEN
              IF m HEAD HEAD 0 >
              THEN "Minimum"
              ELSE "Maximum"
              END
            ELSE "Saddle"
            END
          ELSE DROP "Inconclusive"
          END
        »
        DOLIST p SWAP →TAG v PURGE
      »
    »
  »
```

DATFIT **Find the Best-Fitting Curve for a Data Set** (147)

2216 bytes #C665h

4: Data matrix	4:
3: Column of indep. variable	3: Covariance matrix
2: Column of dep. variable	2: Best-fitting curve
1: Estimate of measurement errors ====>	1: List of labeled coefficients with std. dev's

```
« -3 CF 0 'MARKER' STO 4 ROLLD 3 PICK SWAP COL- OBJ→ 1 GET
  →LIST 4 ROLLD DROP DUP2 COL- OBJ→ 1 GET →LIST 4 ROLLD
  DROP2 OVER SIZE 3 PICK 5 PICK (0,1) * ADD
  IF 6 PICK DUP TYPE
  THEN (0,1) * OBJ→ 1 GET →LIST
  ELSE
    (0,1) * 1 SWAP 4 PICK
    1 SWAP 1 - START DUP ROT 1 + SWAP NEXT
    SWAP →LIST
  END
  « "LEAST-SQUARES FIT" { { "MODEL:" "ENTER GENERAL MODEL
    OF FIT" 9 } { "PARAM:" "ENTER LIST OF PARAMETER NAMES"
    5 } { "VALUES:" "CURRENT VALUES OF PARAMETERS" 5 } {
    "STD.DEV:" "CURRENT STD. DEVIATION FOR MODEL" 5 } } {
    1 3 } { } fields 1 ←nv DOLIST INFORM
  »
  « IF DEPTH DUP ROT EVAL DEPTH 1 - ROT ==
    THEN { NOVAL } 1 GET
    END
    SWAP DROP
  »
  0 DUP DUP
  → w yl xl dat n xe we ←infm ←nv da cm sig
  « IF w TYPE NOT
    THEN
      1 dat SIZE
      1 GET START w SQ INV NEXT
      dat SIZE 1 GET →LIST
    ELSE
      IF w DUP TYPE 3 ==
      THEN
        IF DUP SIZE SIZE 1 ≠
        THEN →DIAG OBJ→ 1 GET →LIST
        ELSE OBJ→ 1 GET →LIST 1 « SQ INV » DOLIST
        END
```

```
         END
      END
      'wl' STO { ↓↑f ↓↑p ↓↑v ↓↑s } DUP 1 « PGALL » DOLIST
      'fields' STO { NOVAL } DUP DUP + + w 1 →LIST +
      fields STO
      WHILE ←infm EVAL
      REPEAT
         EVAL DROP DUP SIZE
         →  m
         «  3 →LIST { ↓↑f ↓↑p ↓↑v } STO ↓↑v ↓↑p STO ↓↑p 1
            « ↓↑f SWAP FDER » DOLIST 'da' STO yl xl 2 « 'x'
            STO ↓↑f EVAL - » DOLIST OBJ→ 1 2 →LIST →ARRY
            1 n
            FOR i
               xl i GET 'x' STO
               1 m FOR j da j GET EVAL NEXT
            NEXT
            { n m } →ARRY wl DUP SIZE SWAP OBJ→ →ARRY
            SWAP DIAG→
            IF wl ΣLIST NOT THEN 1 SF END
            →  ym xm wm
            «  xm TRN wm * xm * INV DUP xm TRN * wm * ym *
               DUP m 1 →LIST RDM 3 ROLLD ym xm ROT * - DUP
               TRN SWAP * 1 GET n m - /
               IF 1 FS?C THEN SWAP OVER * SWAP END
               √ 'x' PURGE ROT OBJ→ 1 GET →LIST ↓↑v ADD
               '↓↑v' STO '↓↑s' STO DUP →DIAG OBJ→ 1 GET
               →LIST √ 'sig' STO 'cm' STO ↓↑v ↓↑p STO ↓↑f
               STEQ { #0h #0h } PVIEW ERASE FUNCTION xl «
               MIN » STREAM ABS NEG 1.1 * xl « MAX » STREAM
               ABS 1.1 * XRNG 'x' INDEP yl « MIN » STREAM
               yl « MAX » STREAM DUP2 - ABS 1.2 * DUP .05 *
               ROT + ABS 3 ROLLD .15 * - ABS NEG SWAP YRNG
               0 RES DRAX DRAW 1 n
               FOR i
                  xe i GET we i GET DUP2 - 3 ROLLD +
                  LINE
               NEXT
               PICTURE
            »
         »
      END
      IF w ↓↑s SAME NOT
      THEN
         ↓↑v ↓↑p STO 'x' PURGE cm ↓↑f EVAL ↓↑p ↓↑v sig 3
         « "±" SWAP + + SWAP →TAG » DOLIST
      END
      VARS DUP 'MARKER' POS 1 SWAP SUB PURGE
   »
»
```

DEFINT **Find Various Approximations of a Definite Integral** (140)

365.5 bytes #DB3Eh

2: Integral expression		2:
1: Number of intervals in approximation	====>	1: list of approximations

```
«  SWAP OBJ→ DROP2 0
```

```
      →  n a b f v d
      «  b a - n /
         →  t
         «  0 1
            FOR j
               0 1 n
               FOR k  a k j + 1 - t * + v STO f →NUM t * +
               NEXT
               .5
            STEP
            v PURGE 3 PICK OVER + 2 / 3 PICK 2 * OVER +
            3 / t LOG ABS 1 + CEIL DUP 2 + 7 ROLLD FIX a b f
            v ∫ →NUM 6 →LIST SWAP RND STD { left mid right
            trap simp intg } SWAP OVER →TAG SWAP PURGE
         »
      »
   »
```

DIRδ Find the Derivative of a Function with Respect to a Path (245)

194 bytes #2D36h

4: Scalar multivariable function		4:
3: Parametrized curve path (symbolic vector)		3:
2: List of variables		2: Symbolic derivative
1: Equation defining parameter and its value	====>	1: Numeric derivative

```
«  -3 CF EQ→
   →  f r v t p
   «  v t + PURGE f v GRADI r v STO 1 « EVAL » DOLIST r t
      VDER DUP 3 ROLLD SDOT SWAP VABS / p t STO →NUM v t +
      PURGE
   »
»
```

EULPLT Plot Euler Estimate of a Differential Equation Solution (172)

432.5 bytes #1054h

5: Symbolic expression of slope		5:
4: List of variables: {indep depnd}		4:
3: Plotting range: {begin end}		3:
2: Initial condition: (x_0,y_0) or list of ordered pairs		2:
1: Number of points for Euler estimate	====>	1:

```
«  -3 CF 4 PICK OBJ→ DROP 5 PICK EVAL - ABS 4 ROLL /
   →  f v p i x y d
   «  IF i DUP TYPE 5 ≠ THEN 1 →LIST END
      1
      «  DUP { } SWAP C→R y STO x STO
         DO
            x →NUM y →NUM R→C + y f d * + →NUM y STO
            x d + →NUM x STO
         UNTIL x p 2 GET >
         END
         SWAP C→R y STO x STO { }
         DO
```

*** *Be sure to read the instructions on pages 283-284 before keying in these programs.* ***

```
            x →NUM y →NUM R→C + y f d * - →NUM y STO
            x d - →NUM x STO
        UNTIL x p 1 GET <
        END
        TAIL REVLIST SWAP + 2 « LINE » DOSUBS
        v PURGE DRAX
      »
      DOLIST PICTURE
    »
»
```

EXCO — Expand and Collect Completely (201)

65.5 bytes #BB88h

1: symbolic expression	====>	1: revised expression

```
« « EXPAN » MULTI « COLCT » MULTI »
```

FDER — Compute a Formal (Symbolic) Integral (68)

139 bytes #17B2h

2: Function		2:
1: Variable of differentiation	====>	1: Formal derivative

```
«  -3 CF DUP
   →  x ↑↓
   «  IF DUP TYPE 9 == THEN x SHOW END
      x { ↑↓ } + ↓MATCH DROP '↑↓' DUP STO ↑↓ ∂ x '↑↓' STO
      EVAL
   »
»
```

FUNOF? — Is an Expression a Function of a Particular Variable? (175)

61 bytes #DC10h

2: Expression		2:
1: Variable	====>	1: Flag 0/1

```
«  → f v « f v DUP 2 →LIST ↓MATCH SWAP DROP » »
```

FVP — Fundamental Vector Product at Point on Parametrized Surface (256)

126 bytes #8FA8h

3: Parametrized surface (symbolic vector		3:
2: List of parameters		2:
1: List of parameter values at point	====>	1: F.V.P. (as a symbolic vector)

```
«  →  f v p
   «  f v HEAD VDER f v 2 GET VDER p v STO SCROSS
```

```
      1 « →NUM » DOLIST
    »
»
```

gcd Find the Greatest Common Divisor of Two Numbers (300)

45 bytes #7CB5h

2: number 1		2:
1: number 2	====>	1: greatest common divisor

```
«  WHILE DUP REPEAT SWAP OVER MOD END
   DROP ABS
»
```

GRADI Compute the Gradient of a Scalar Function (241)

80.5 bytes #C6CCh

2: Scalar function		2:
1: List of variables	====>	1: Gradient (as symbolic vector)

```
«  -3 CF → f v «v 1 « f SWAP FDER » DOLIST » »
```

IMPδ Compute the Implicit Derivative of a Relation (75)

497 bytes #FBA1h

2: Relation expression		2:
1: List of implicit variables, differentiation var. listed 1st	====>	1: Derivative

```
«  -3 CF DUP 1 GET
   → f s v
   «  s 1
      «  IF DUP v SAME
         THEN '↑↓' 2 →LIST '↓x' STO
         ELSE
            DUP "'↑" SWAP + DUP "(↑↓)'" + OBJ→ SWAP OBJ→
            "'der↑" 4 PICK + "(↑↓,1)" + OBJ→ "'δ" 5 PICK +
            "." + v + "'" + OBJ→ 5 →LIST
         END
      »
      DOLIST '↓y' STO f ↓x ↓MATCH DROP 1 ↓y SIZE
      FOR k ↓y k GET 1 2 SUB ↓MATCH DROP NEXT
      '↑↓' ∂ 1 ↓y SIZE
      FOR k
         ↓y k GET DUP 2 3 SUB ROT SWAP ↓MATCH DROP SWAP 4
         5 SUB ↓MATCH DROP
      NEXT
      ↓x REVLIST ↓MATCH DROP 1 ↓y SIZE
      FOR k ↓y k GET DUP 3 3 SUB SWAP 1 GET + ↓MATCH DROP
      NEXT
      { ↓x ↓y } PURGE
   »
»
```

INDEF Compute an Indefinite Integral (132)

660.5 bytes #2500h

2: Function		2: Indefinite integral
1: Variable of integration	====>	1: 1/0 (success or failure)

or

2:		2: Indefinite integral
1: Symbolic integral expression	====>	1: 1/0 (success or failure)

```
« -3 CF
  IF DUP TYPE 9 ==
  THEN OBJ→ DROP2 ROT DROP ROT DROP
  END
  DUP
  → f ↑↓ w
  « IF f TYPE NOT THEN '0*x' f + 'f' STO END
    1 ↑↓ f ↑↓ SHOW '↑↓' DUP STO OVER ∫
    IF DUP OBJ→ DUP { ∫ } 1 GET SAME
    THEN DROP DROPN 0
    ELSE
       5 ROLL DROP
       IF { + } 1 GET SAME
       THEN
          →LIST 0 SWAP 1
          « IF DUP { '∫(&1,&2,&3,&4)' ↑↓ } ↓MATCH
            THEN
               DROP COLCT EVAL
               IF DUP { '∫(&1,&2,&3,&4)' ↑↓ } ↓MATCH
               THEN DROP SWAP
               ELSE
                  DROP OBJ→ 3 DROPN EVAL '↑↓' w 2 →LIST
                  ↓MATCH DROP SWAP .5 +
               END
            ELSE
               DROP OBJ→ 3 DROPN EVAL '↑↓' w 2 →LIST
               ↓MATCH DROP SWAP .5 +
            END
          »
          DOLIST OBJ→ 1 + ROLLD + SWAP
       ELSE DROP2 EVAL '↑↓' w 2 →LIST ↓MATCH DROP 1
       END
    END
    IF DUP .5 == THEN .5 - END
  »
»
```

ΣINT1 Integral of Data Set (Piecewise Linear Interpolation) (143)

251.5 bytes # B16h

3: Data matrix		3:
2: Column of indep. variable		2:
1: Column of dep. variable	====>	1: Definite integral

```
« 3 PICK SWAP COL- OBJ→ 1 GET →LIST 4 ROLLD DROP COL- OBJ→
```

```
1 GET →LIST 3 ROLLD DROP OVER SORT DUP SIZE 0
→ u v us n s
« 1 n 1 -
  FOR k
    us k GETI 3 ROLLD GET SWAP 2 →LIST DUP 1 « v u
    ROT POS GET » DOLIST EVAL + 2 / SWAP EVAL - * s +
    's' STO
  NEXT
  s
»
```

ΣINT2 Integral of Data Set (Single Polynomial Interpolation) (145)

217 bytes #BE0Ch

4: Data matrix		4:	
3: Column of indep. variable		3:	
2: Column of dep. variable		2:	
1: List with limits of integration	====>	1:	Definite integral

```
« EVAL 5 ROLLD 5 ROLLD 3 PICK SWAP COL- 4 ROLLD DROP COL-
  3 ROLLD DROP 2 COL→ PFIT OBJ→ 1 GET
  → n
  « 1 n FOR k k / n ROLLD NEXT
    0 n 1 + →ARRY
  »
  'x' P→SYM DUP ROT 'x' STO →NUM ROT 'x' STO SWAP →NUM -
  'x' PURGE
»
```

ΣINT3 Integral of Data Set (Cubic Spline Interpolation) (146)

943.5 bytes #F210h

3: Data matrix		3:	
2: Column of indep. variable		2:	
1: Column of dep. variable	====>	1:	

```
« 3 PICK SWAP COL- OBJ→ 1 GET →LIST 4 ROLLD DROP COL- OBJ→
  1 GET →LIST 3 ROLLD DROP OVER SORT DUP SIZE .0001 10
  -1.E30 DUP 0 DUP DUP2 DUP
  → u v us n eps jmax os ost j it s st del
  « us 1 « v u ROT POS GET » DOLIST DUP DUP 1 GET SWAP n
    GET us DUP 1 GET SWAP n GET us 6 ROLL n OVER 1 2 SUB
    EVAL SWAP - us 1 2 SUB EVAL SWAP - / 3 PICK n DUP 1
    - SWAP SUB EVAL SWAP - us n DUP 1 - SWAP SUB EVAL
    SWAP - / 2 →LIST SPLINE ROT DROP 0 DUP
    → fa fb a b vs ys x sum
    « DO
        IF 'j' INCR 1 ==
        THEN b a - fa fb + * 2 / 1
        ELSE
          b a - it / 'del' STO a del 2 / + 'x' STO
          0 'sum' STO 1 it
          START
            us vs ys x SPLEVAL sum + 'sum' STO x del
```

```
                + 'x' STO
              NEXT
              st b a - sum * it / + 2 / it 2 *
            END
            'it' STO 'st' STO
          UNTIL
            st 4 * ost - 3 / DUP 's' STO os - ABS os
            ABS eps * < j jmax == OR s 'os' STO st 'ost'
            STO
          END
          s
        »
      »
    »
```

ΣINT4 Integral of a Data Set (Using Fitted Curve) (148)

89.5 bytes #4041h

5: Data matrix		5:
4: Column of indep. variable		4:
3: Column of dep. variable		3:
2: Estimate of measurement errors		2:
1: List of limits of integration	====>	1: Definite integral

```
«  EVAL
   →  ↑↓a ↑↓b
   «  DATFIT DROP SWAP DROP ↑↓a ↑↓b ROT 'x' ∫ →NUM
   »
»
```

INTEVAL Evaluate Integrand before Evaluating Integral (298)

234.5 bytes #624Fh

1: symbolic integral	====>	1: Evaluated integral

```
«  -3 CF DUP
   →  fn fx
   «  «  → f1 f2 f3 f4
         «  f1 f2 f3 f4 ∫
         »
      »
      'fx' STO fn
      { '∫(&1,&2,&3,&4)' 'fx(&1,&2,&3,QUOTE(&4))' }
      ↓MATCH DROP EVAL
   »
»
```

IPARTS — Integrate by Parts (186)

264 bytes #84F9h

3: u		3:
2: dv		2:
1: variable of integration (x)	====>	1: Symbolic expression

```
«  →  u dv x
   «  x PURGE u x FDER dv x INDEF DROP 0 'ç' STO
      EVAL DUP u * COLCT 3 ROLLD * COLCT x INDEF
      IF 'ç' 4 ROLLD
      THEN EVAL - SWAP +
      ELSE
         EVAL COLCT OBJ→ DROP2 ROT DROP ROT DROP INDEF
         IF THEN EVAL - SWAP + ELSE EVAL - SWAP DROP END
      END
      'ç' PURGE
   »
»
```

JACOB — Compute the Jacobian Matrix of a Vector Function (252)

116 bytes # 9A6h

2: symbolic vector field		2:
1: list of variables	====>	1: Symbolic Jacobian matrix

```
«  →  f x
   «  f 1
      «  1 x SIZE FOR k DUP x k GET FDER SWAP NEXT
         DROP x SIZE →LIST
      »
      DOLIST
   »
»
```

LAPLC — Compute the Laplacian of a Function (254)

213 bytes #1DC7h

2: Function (scalar or vector)		2:
1: List of variables	====>	1: Laplacian

```
«  IF SWAP DUP TYPE 5 ≠ THEN 1 →LIST END
   DUP SIZE
   →  v f n
   «  f 1 « v GRADI » DOLIST 1 n
      START v NEXT
      n →LIST 2 « 2 « FDER » DOLIST ΣLIST COLCT » DOLIST
      IF n 1 == THEN EVAL END
   »
»
```

lcm Find the Least Common Multiple of Two Numbers (310)

34 bytes #CFB8h

2: number 1	2:
1: number 2 ====>	1: least common multiple

```
«  DUP2 gcd / * ABS »
```

LIM Search for the Limit of an Expression (24)

355.5 bytes #FD40h

3: limit expression list	3:
2: starting index magnitude for search	2:
1: ending index magnitude for search ====>	1: list of sampled values toward limit

```
«  ROT OBJ→ DROP OBJ→ DROP2 5 ROLL DUP SIGN SWAP ABS 0
   →  t f v l s b d
   «  CLLCD 0 b t
      FOR j
         j ALOG
         IF l TYPE 9 ≠
         THEN INV s * l +
         ELSE IF l →NUM IM SIGN -1 == THEN NEG END
         END
         →NUM v STO f →NUM DUP 1 DISP
         IF
            j 10 * t b - / DUP IP ≠ SWAP DUP 4 PICK == ROT
            OR
         THEN DROP
         ELSE d 1 + 'd' STO
         END
         t b - 100 /
      STEP
      d →LIST SWAP DROP v PURGE
   »
»
```

LININ? Is Given Expression Linear in Given Variable? (175)

603 bytes #B895h

2: Expression	2:
1: Variable ====>	1:

```
«  →  e x
   «  0 0 1 0 0
      →  args lin go non occ
      «  «  1 SWAP START lin EVAL NEXT »
         'args' STO
         «  →  sube
            «  IF go
               THEN
                  CASE sube TYPE 9 SAME
```

```
                    THEN sube OBJ→
                       CASE
                          OVER NOT THEN DROP2 END
                          { NEG + - * / } SWAP POS DUP NOT
                             THEN
                                DROP 1 → non « args EVAL »
                             END
                          DUP 4 < THEN DROP args EVAL END
                          SWAP DROP 5 SAME
                             THEN
                                1 → non « lin EVAL »
                                lin EVAL
                             END
                          occ SWAP lin EVAL occ DUP ROT -
                          ROT lin EVAL occ ROT -
                          IF AND THEN 0 'go' STO END
                       END
                    END
                 sube x SAME
                    THEN
                       1 'occ' STO+
                       IF non THEN 0 'go' STO END
                    END
                 END
              END
           »
        »
        'lin' STO e lin EVAL go occ AND
     »
  »
»
```

LN∂ Perform Logarithmic Differentiation (81)

263.5 bytes #2D31h

3: list of numerator factors		3:
2: list of denominator factors		2:
1: variable of differentiation	====>	1: Symbolic logarithmic derivative

```
«  →  n d v
   «  n d 2 →LIST 1
      «  IF DUP SIZE 1 > THEN πLIST ELSE 1 GET END
      »
      DOLIST EVAL / n 1 « LN v FDER » DOLIST
      d 1 « LN v FDER » DOLIST 2 →LIST 1
      «  IF DUP SIZE 1 > THEN ΣLIST ELSE 1 GET END
      »
      DOLIST EVAL - * COLCT
   »
»
```

LTRIM Trim Zero Elements from the Left End of a Polynomial (305)

135.5 bytes #200Ah

1: polynomial	====>	1: trimmed polynomial

```
«  DUP RNRM
   IF .00001 <
   THEN DROP 0 1 →ARRY
   ELSE
      OBJ→ 1 GET 1 +
      WHILE DUP ROLL DUP ABS .00001 <
      REPEAT DROP 1 -
      END
      OVER ROLLD 1 - →ARRY
   END
»
```

MAXMIN Find the Maximum or Minimum within the Given Interval (98)

459 bytes #1F54h

4: Function		4:
3: Independent variable		3:
2: Max/min and integer/decimal code		2:
1: { begin end } search interval	====>	1: Maximum or minimum

```
«  →  f v s ab
   «  f v FDER
      IF s TYPE 0 ≠
      THEN
         DUP →RPN 1
         «  IF DUP TYPE 6 ≠ THEN DROP END
         »
         DOLIST PURGE v QUAD
      ELSE
         v ab ROOT
         IF s ABS 1 ==
         THEN
            DUP FLOOR SWAP CEIL 2 →LIST DUP 1
            « v STO f EVAL » DOLIST DUP
            IF s 0 < THEN « MIN » ELSE « MAX » END
            STREAM POS GET
         END
         v →TAG f EVAL EVAL
         IF DUP TYPE
         THEN
            →RPN 1
            «  IF DUP DUP TYPE 6 == SWAP { = } HEAD SAME OR
               THEN DROP
               END
            »
            DOLIST EVAL →NUM
         END "f(" v + ")" + →TAG v PURGE
      END
   »
»
```

MIDPT Midpoint Estimate of Integral (Open Interval Romberg) (305)

452 bytes #6094h

5: Function		5:
4: Variable		4:
3: Lower limit		3:
2: Upper limit		2:
1: Number of partitions	====>	1: Estimated integral

```
« → f v a b n
   « IF n 1 ==
     THEN b a - a b + .5 * v STO f EVAL * 1
     ELSE
        b a - 3 it * / DUP 2 * OVER .5 * a + 0
        → del ddel x sum
        « 1 it
          FOR p
             x v STO f EVAL sum + 'sum' STO x ddel +
             'x' STO x v STO f EVAL sum + 'sum' STO x del
             + 'x' STO
          NEXT
          r b a - sum * it / + 3 / it 3 *
        »
     END
     'it' STO 'r' STO
   »
»
```

MLTINT Find Numeric Multiple Integral (258)

280.5 bytes #1552h

1: Symbolic, nested, multiple integral expression	====>	1: numeric integral

```
« -3 CF →RPN 1
   « IF DUP { ∫ } 1 GET SAME
     THEN DROP « ∫ EVAL EVAL EXCO »
     END
   »
   DOLIST EVAL
   IFERR
     IF DUP →RPN DUP SIZE GET { ∫ } 1 GET SAME
     THEN
        IF
           OBJ→ DROP2 OVER { '∫(&1,&2,&3,&4)' } 3 PICK +
           ↓MATCH
        THEN DROP 4 FIX
        ELSE DROP STD
        END
        ∫
     ELSE STD
     END
     →NUM
   THEN DROP
   END
»
```

MULTI — Repeat Given Function until Result is Same as Input (294)

56 bytes #860Ah

1: object	====>	1: revised object

```
«  →  p « DO DUP p EVAL DUP ROT UNTIL SAME END » »
```

NLSYS — Solve a System of Non-Linear Equations (108)

571 bytes #63D6h

3: List of equations		3:
2: List of variables		2:
1: List of starting guesses	====>	1: list of solutions

```
«  -22 SF
   IF DUP TYPE 1 == THEN C→R 2 →LIST END
   3 ROLLD DUP2 JACOB 0
   →  s f x j n
   «  IF f SIZE DUP x SIZE == SWAP s SIZE == AND
      THEN
         DO
            'n' INCR s DUP x STO OBJ→ →ARRY j 1
            « 1 « →NUM » DOLIST » DOLIST SM→ 2 →LIST
            →ARRY INV f 1 « →NUM » DOLIST OBJ→ →ARRY * -
         UNTIL
            s OBJ→ →ARRY ABS OVER ABS - ABS .000000000001
            ≤ f 1 « →NUM » DOLIST ΣLIST ABS .000000000001 ≤
            OR SWAP OBJ→ 1 GET →LIST 's' STO SWAP 20 ≥ OR
         END
         IF n 20 <
         THEN s 1 « -10 RND » DOLIST x →TAG
         ELSE "No solution found" MSGBOX { }
         END
         x PURGE
      ELSE # 501h DOERR
      END
   »
»
```

NSEGINT — Numerically Segment a Definite Integral (160)

163 bytes #8482h

2: Integral expression		2:
1: Value of indep. variable at pt. of segmentation	==>	1: numerically evaluated integral

```
«  →  ↑↓
   «  { '∫(&1,&2,&3,&4)' '∫(&1,↑↓,&3,&4)+∫(↑↓,&2,&3,&4)' }
      ↓MATCH DROP →NUM
   »
»
```

OPINT Find an Improper Integral—Open Interval (167)

645 bytes #2B9Ch

1: Improper integral expression ====> 1: Evaluated integral

```
« OBJ→ DROP2 4 ROLL 4 ROLL .000001 14 5 { 1 } DUP 1
  → f v a b eps jmax k s h j
  « IF f v INDEF
    THEN
       -22 SF v PURGE XRCONV 'f' STO b v STO f EVAL a v
       STO f EVAL - COLCT →NUM v PURGE -22 CF
    ELSE
       DROP 0 0
       DO
          DROP2 f v a →NUM b →NUM j MIDPT
          IF j k ≥
          THEN
             s OBJ→ →ARRY 1 h SIZE
             FOR i
                h i GET h SIZE 1 - 0
                FOR m DUP m ^ SWAP -1 STEP
                DROP
             NEXT
             h SIZE DUP 2 →LIST →ARRY SWAP DUP2 OVER
             / DUP 5 ROLLD RSD SWAP / 0 PEVAL SWAP 0
             PEVAL SWAP
          ELSE 1 1
          END
       UNTIL
          DUP ABS 3 PICK ABS eps * < jmax j == OR s r
          + 's' STO h DUP j GET 9 / + 'h' STO 'j' INCR
          DROP
       END
       DROP
    END
    { r it } v + PURGE
  »
»
```

PADD Perform Polynomial Addition (76)

172.5 bytes #64CBh

2: [polynomial 1]		2:	
1: [polynomial 2]	====>	1:	[polynomial 1 + polynomial 2]

```
« RCLF 3 ROLLD -55 CF
  IFERR +
  THEN
     OVER SIZE 1 GET OVER SIZE 1 GET -
     IF DUP 0 < THEN ABS ROT SWAP END
     → a d
     « 1 d START 0 NEXT
       a OBJ→ 1 GET d + →ARRY +
     »
  END
  LTRIM SWAP STOF
»
```

PAPROX Compute Approximation after Plotting Taylor Series (46)

147 bytes #295Eh

5: function being approximated	5:
4: independent variable	4:
3: order of Taylor series desired	3: Taylor series
2: point around which approx. is centered	2: Approximate value
1: point being approximated ====>	1: Estimated error

```
«  5 ROLLD 4 DUPN TYLRa
   → b f v o a p
   «  p f v a b PTAYL b v STO p DUP →NUM f →NUM OVER - ABS
      v PURGE
   »
»
```

PARAδ Find Derivative of Function described Parametrically (83)

101 bytes #2ED1h

2: List of parametric definitions	2: Symbolic derivative
1: Value of t at which to compute slope ====>	1: Slope at given t

```
«  't' STO 1 « 't' FDER COLCT » DOLIST
   DUP EVAL SWAP / COLCT DUP →NUM 't' PURGE
»
```

PATHINT Compute a Path Integral (273)

216.5 bytes # 2h

5: Function	5:
4: Parametrized curve (as symbolic vector)	4:
3: List of function variables	3:
2: Parameter variable	2:
1: List of starting and ending parameter values ====>	1: Path integral

```
«  -3 CF EVAL 6 ROLLD 6 ROLLD
   → f c x t
   «  c t VDER f
      IF DUP TYPE 5 ≠ THEN 1 →LIST END
      c x STO 1 « EVAL COLCT » DOLIST
      IF DUP SIZE 1 >
      THEN SDOT
      ELSE SWAP VABS * EVAL
      END
      COLCT t ∫ →NUM x PURGE
   »
»
```

PCONV Convert an Expression to a Polynomial (76)

682.5 bytes #27D6h

1: expression	====> 1: polynomial

```
« -3 CF { [ 0 ] [ 1 ] } { N D } STO COLCT →RPN DUP SIZE
  « → n d
    « N d PMULT D n PMULT 'OP' EVAL 'N' STO
      D d PMULT 'D' STO
    »
  »
  → p n ←pdiv
  « 1 n
    FOR k
       'p' k GET
       IF DUP TYPE
       THEN
          IF DUP TYPE { 6 7 } SWAP POS
          THEN DROP [ 1 0 ] 1 →LIST
          ELSE
             { + - * ^ NEG DEC } SWAP POS
             IF DUP
             THEN
                { PADD PSUB PMULT PPOWER PSUB
                « EVAL DUP DROP » } SWAP GET 1 →LIST
             ELSE
                DROP p k 1 + GET { + - } SWAP POS
                { PADD PSUB } SWAP GET 'OP' STO
                { ←pdiv DEC }
             END
          END
       ELSE 1 →ARRY 1 →LIST
       END
       p k ROT REPL 'p' STO
    NEXT
    p EVAL D PMULT N PADD D { OP N D } PURGE
  »
»
```

PDER Find the Derivative of a Polynomial (76)

112.5 bytes #1DFFh

1: Polynomial	====> 1: Derivative of Polynomial

```
« OBJ→ 1 GET 1 -
  → n
  « DROP
    IF n
    THEN
       1 n FOR k k * n ROLLD NEXT
       n
    ELSE 0 1
    END
    →ARRY
  »
»
```

PDIV Perform Polynomial Division (Euclidean) (76)

336 bytes #1FDFh

3:			3: Quotient of P1/P2
2: Polynomial 1			2: Numerator of remainder
1: Polynomial 2	====>		1: Denominator of remainder

```
«  LTRIM DUP OBJ→ OBJ→ DROP →LIST ROT LTRIM OBJ→ OBJ→ DROP
   →LIST SWAP DUP2 SIZE SWAP SIZE
   IF OVER - DUP 0 <
   THEN 3 DROPN [ 0 ] 3 ROLLD OBJ→ →ARRY SWAP OBJ→ →ARRY
   ELSE
      SWAP ROT DUP 1 GET
      →  n p2 t
      «  { } 3 ROLLD 0 SWAP
         START
            DUP 1 GET t ⁄ ROT OVER + 3 ROLLD 1 n
            FOR d
               OVER d GET p2 d GET 3 PICK * - ROT d ROT
               PUT SWAP
            NEXT
            DROP 2 OVER SIZE MIN 1.E499 SUB
         NEXT
      »
      SWAP OBJ→ →ARRY SWAP OBJ→ →ARRY ROT
   END
»
```

PDIV2 Perform Division of a Reduced Polynomial (76)

189 bytes #6A52h

3:			3: Quotient of P1/P2
2: Polynomial 1			2: Numerator of remainder
1: Polynomial 2	====>		1: Denominator of remainder

```
«  →  p q
   «  q PFACT
      IF DUP SIZE 1 >
      THEN
         1
         «  p SWAP PDIV SWAP
            IF ABS NOT
            THEN q SWAP PDIV DROP2 'q' STO 'p' STO
            ELSE DROP2
            END
         »
         DOLIST
      END
      p q PDIV
   »
»
```

Compute Partial Fractions from a Rational Fraction (76)

1258 bytes #F862h

2: Numerator polynomial		2:	
1: Denominator polynomial	====>	1: List of partial fractions	

```
« → p
  « [ 1 ] p SIZE 1 SWAP
    FOR k p k GETI 3 ROLLD GET 1 →ARRY PPOWER PMULT 2
    STEP
    PDIV ROT 'x' P→SYM { } + 3 ROLLD
    → ←n ←d
    « 1 p SIZE
      FOR k
        p k GETI 'x' P→SYM 3 ROLLD GET DUP 1
        FOR j 2 OVER + j - PICK j ^ SWAP -1 STEP
        →LIST SWAP DROP p k GETI 3 ROLLD GET
        → r m
        « r m 1 →ARRY PPOWER ←d OVER PDIV DROP2
          SWAP ←n 1 CF 3 ROLLD DUP2 SIZE SWAP SIZE
          IF > 1 GET THEN 1 SF SWAP END
          [ 0 ] [ 1 ] DUP2
          → c a b u v y x
          « WHILE a b PDIV DROP DUP ABS 8 RND
            REPEAT
              b 'a' STO 'b' STO u DUP 3 PICK
              PMULT NEG x PADD 'u' STO 'x' STO v DUP
              ROT PMULT NEG y PADD 'v' STO 'y' STO
            END
            DROP2 c b PDIV DROP
            IF ABS
            THEN DROP "No solution"
            ELSE
              u OVER PMULT v ROT PMULT
              IF 1 FS?C THEN SWAP END
            END
          »
          DROP 1 m
          START r PDIV DROP SWAP NEXT
          DROP m →LIST 1
          « IF DUP SIZE 1 GET 1 ==
            THEN [ 0 1 ] PMULT
            END
          »
          DOLIST 2
          « A→Q
            → z y
            « 1 2
              FOR q
                IF y DUP q GET ABS .00001 <
                THEN q 0 PUT
                END
                IF q GET DUP TYPE
                THEN →RPN REVLIST TAIL
                  IF DUP SIZE 1 ==
                  THEN EVAL 2 SF 1
                  ELSE
                    IF DUP SIZE 3 ==
                    THEN TAIL 2 SF
```

```
                        END EVAL SWAP
                     END
                  ELSE 1
                  END
               NEXT
               SWAP 4 ROLL SWAP 2 →LIST 3 ROLLD 2
               →LIST DUP EVAL lcm DUP ROT / ROT *
               { x 1 } * ΣLIST SWAP z * /
               IF 2 FS?C THEN NEG END
            »
         »
         DOLIST
      »
      + 2
   STEP
   ΣLIST
  »
 »
»
```

PFACT — Factor a Polynomial (76)

737 bytes #4922h

1: polynomial	====>	1: list of polynomial factors

```
« 0
  → p t
  « p OBJ→ 1 GET →LIST « gcd » STREAM
    IF DUP 1 ≠
    THEN { 1 } →ARRY 1 →LIST
    ELSE DROP { }
    END
    p RROOTS { }
    → fct r ifct
    « 1 r SIZE
      FOR k
        r k GET
        IF DUP TYPE DUP 1 + 't' STO
        THEN r k 1 + GET 2 →ARRY
        ELSE 1 →ARRY
        END
        PCOEF P→Qc
        IF DUP DUP ABS SWAP 0 RND ABS - ABS .00001 <
        THEN 0 RND fct SWAP + 'fct' STO
        ELSE
          IF t 1 -
          THEN ifct SWAP + 'ifct' STO
          ELSE DROP
          END
        END
        t
      STEP
      1 ifct SIZE
      IF DUP
      THEN
        FOR j
          ifct j GETI 3 ROLLD GET PMULT P→Qc
          IF DUP DUP ABS SWAP 0 RND ABS - ABS .00001 <
```

```
            THEN 0 RND fct SWAP + 'fct' STO
            ELSE DROP
            END
          NEXT
      ELSE DROP2
      END
      fct DUP { [ 1 ] [ 1 ] } + « PMULT » STREAM p
      SWAP PDIV DROP2
      IF DUP [ 1 ] ≠ THEN + ELSE DROP END
    »
  »
»
```

PFIT Interpolate a Data Set with Single Polynomial (145)

205.5 bytes **#AF57h**

1: Data matrix	====>	1: Interpolating polynomial (vector form)

```
« DUP SIZE 1 GET
  → a s
  « 1 s
    FOR j a { j 2 } GET NEXT
    s →ARRY 1 s
    FOR j
      a { j 1 } GET s 1 - 0
      FOR k DUP k ^ SWAP -1 STEP
      DROP
    NEXT
    { s s } →ARRY SWAP DUP2 OVER / DUP 5 ROLLD RSD
    SWAP / +
  »
»
```

PFRAC Compute Partial Fractions from a Symbolic Expression (191)

1389 bytes **#B5A0h**

1: Expression	====>	1: List of partial fractions

```
« S→RF PDIV ROT 'x' P→SYM { } + 3 ROLLD
  → ←n ←d
  « ←d PFACT 0 RND { }
    → f c
    « f 1
      « IF DUP c SWAP POS
        THEN DROP
        ELSE
          0 SWAP f +
          « IF DUP2 SAME
            THEN ROT 1 + 3 ROLLD DROP
            ELSE DROP
            END
          »
          STREAM SWAP OVER c + 'c' STO
        END
      »
```

```
    DOLIST
  »
  → p
  « 1 p SIZE
    FOR k
        p k GETI 'x' P→SYM 3 ROLLD GET DUP 1
        FOR j 2 OVER + j - PICK j ^ SWAP -1 STEP
        →LIST SWAP DROP p k GETI 3 ROLLD GET
        → r m
        « r m 1 →ARRY PPOWER ←d OVER PDIV DROP2
          SWAP ←n 1 CF 3 ROLLD DUP2 SIZE SWAP SIZE
          IF > 1 GET THEN 1 SF SWAP END
          [ 0 ] [ 1 ] DUP2
          → c a b u v y x
          « WHILE a b PDIV DROP DUP ABS 8 RND
            REPEAT
                b 'a' STO 'b' STO u DUP 3 PICK
                PMULT NEG x PADD 'u' STO 'x' STO v DUP
                ROT PMULT NEG y PADD 'v' STO 'y' STO
            END
            DROP2 c b PDIV DROP
            IF ABS
            THEN DROP "No solution"
            ELSE
                u OVER PMULT v ROT PMULT
                IF 1 FS?C THEN SWAP END
            END
          »
          DROP 1 m
          START r PDIV DROP SWAP NEXT
          DROP m →LIST 1
          « IF DUP SIZE 1 GET 1 ==
            THEN [ 0 1 ] PMULT
            END
          »
          DOLIST 2
          « A→Q
            → z y
            « 1 2
                FOR q
                    IF y DUP q GET ABS .00001 <
                    THEN q 0 PUT
                    END
                    IF q GET DUP TYPE
                    THEN
                        →RPN REVLIST TAIL
                        IF DUP SIZE 1 ==
                        THEN EVAL 2 SF 1
                        ELSE
                            IF DUP SIZE 3 ==
                            THEN TAIL 2 SF
                            END EVAL SWAP
                        END
                    ELSE 1
                    END
                NEXT
                SWAP 4 ROLL SWAP 2 →LIST 3 ROLLD 2
                →LIST DUP EVAL lcm DUP ROT / ROT *
                { x 1 } * ΣLIST SWAP z * /
                IF 2 FS?C THEN NEG END
            »
```

```
            »
            DOLIST
          »
          + 2
        STEP
        IF DUP HEAD 0 SAME THEN TAIL END
      »
    »
  »
```

PGALL Purge All Instances of a Variable in the Current Path (65)

92 bytes #68E4h

1: variable name	====>	1:

```
« PATH
  → name path
    « DO name name PURGE EVAL UPDIR UNTIL TYPE 6 == END
      path EVAL
    »
»
```

PLTDER Plot a Higher-Order Derivative Directly (62)

172 bytes #DC7Eh

4: Function		4:
3: Variable of differentiation		3:
2: Order of derivative		2:
1: 0/1: don't erase/erase	====>	1:

```
« -3 CF
  → f v o s
  « v PGALL
    WHILE o 0 > REPEAT f v ∂ 'f' STO o 1 - 'o' STO END
    f STEQ FUNCTION v INDEP AUTO
    IF s THEN ERASE END
    DRAX DRAW PICTURE
  »
»
```

PMULT Perform Polynomial Multiplication (76)

202.5 bytes #44DDh

2: [Polynomial 1]		2:
1: [Polynomial 2]	====>	1: [P1 x P2]

```
« DUP SIZE 1 GET
  → a na
  « DUP DUP 0 CON DUP SIZE 1 GET DUP na + 1 - 1 →LIST
    → b c nb nab
    « { 1 } nb + RDM 1 nb
      START a OBJ→ DROP c OBJ→ DROP NEXT
```

```
      nb DROPN nb nab + →ARRY * nab RDM
    »
  »
»
```

POL∂ Find the Derivative of a Polar Function (84)

167.5 bytes #2DB5h

2: Polar function		2: Symbolic derivative
1: Value of θ to use to compute slope	====>	1: Slope at given θ

```
« 'θ' DUP 4 ROLLD STO DUP ROT FDER COLCT
  → r dr
  « { 'COS(θ)' 'SIN(θ)' } 1 « DUP dr * SWAP r * COLCT »
    DOLIST EVAL 3 ROLLD + 3 ROLLD - / DUP →NUM
  »
»
```

POTEN Find a Potential of a Vector Function (275)

287.5 bytes #68C7h

2: Symbolic vector function		2:
1: List of variables	====>	1: Scalar potential (evaluated as far as possible)

```
« 1
  → f v n
  « f v 2
    « INDEF DROP 1 'n' STO
      WHILE OBJ→ DUP { + - } SWAP POS
      REPEAT DROP2 1 n + 'n' STO
      END
      SWAP DROP EVAL n →LIST
    »
    DOLIST 1 « EVAL » DOLIST COLCT DUP 1 OVER SIZE
    FOR k
      DUP TAIL SWAP HEAD
      IF OVER SWAP POS DUP
      THEN k + ROT SWAP 0 PUT SWAP
      ELSE DROP
      END
    NEXT
    DROP ΣLIST COLCT
  »
»
```

PPOWER Take a Positive Integral Power of a Polynomial (76)

175 bytes #8CFAh

2: [polynomial]		2:
1: [positive integral power]	====>	1: [computed polynomial]

```
«  1 GET
   →  p n
   «  1 DUP →ARRY
      WHILE n 0 >
      REPEAT
         IF n 2 MOD THEN p PMULT END
         n 2 / FLOOR 'n' STO
         IF n THEN p DUP PMULT 'p' STO END
      END
   »
»
```

PPROD Take the Derivative of the Product of a List of Polynomials (76)

57.5 bytes #8C2Ch

1: { list of [polynomials] } ====> 1: derivative of product in polynomial form

```
«  OBJ→ 1 SWAP 1 - FOR k PMULT NEXT PDER »
```

P→Qc Convert the Simple Rational Coefficients in a Polynomial (310)

230.5 bytes #FFCAh

1: polynomial array	====>	1: converted polynomial

```
«  DUP SIZE 1 GET
   →  p n
   «  1 n
      FOR k
         p k GET DUP SIGN SWAP ABS
         IF DUP DUP IP ≠
         THEN
            →Qc
            IF DUP TYPE 9 ==
            THEN
               OBJ→ DROP2
               IF DUP 10 <
               THEN p * k ROT 4 ROLL * PUT 'p' STO
               ELSE 3 DROPN
               END
            ELSE DROP2
            END
         ELSE DROP2
         END
      NEXT
      p
   »
»
```

*** *Be sure to read the instructions on pages 283-284 before keying in these programs.* ***

PQUOT Take the Derivative of the Quotient of Two Polynomials (76)

414.5 bytes #F865h

2: [numerator polynomial, p]		2: [numerator polynomial of deriv.]
1: [denominator polynomial, q]	====>	1: [denominator polynomial of deriv.]

```
«  → p q
   « q p PDER PMULT p q PDER PMULT PSUB PFACT [ 1 ] + q
     [ 2 ] PPOWER PFACT [ 1 ] +
     → n d
     « 1 d SIZE
       FOR k
          IF n d k GET POS DUP
          THEN
             n SWAP [ 1 ] PUT 'n' STO d k [ 1 ] PUT
             'd' STO
          ELSE DROP
          END
       NEXT
       n d 2 →LIST 1 « « PMULT » STREAM » DOLIST EVAL
     »
   »
»
```

PREDUCE Reduce the Coefficients of a Polynomial (76)

66 bytes #652Eh

1: [polynomial]	====>	1: [reduced polynomial]

```
« DUP OBJ→ 1 GET →LIST 0 RND « gcd » STREAM / »
```

PSEQ Plot a Recursively-Defined Sequence (16)

265 bytes #4AD9h

4: recursive sequence definition		4:
3: list of initial values		3:
2: beginning of plot range		2:
1: end of plot range	====>	1:

```
« -3 CF
  → f a s t
  « f a t NEG →SEQ 'seqnc' STO -1 t XRNG seqnc s GET 2 -
    seqnc t GET 2 + YRNG 'n' s t 3 →LIST INDEP 1 RES {
    10 1 } ATICK FUNCTION « seqnc n GET » STEQ ERASE DRAX
    DRAW PICTURE { EQ n seqnc } PURGE
  »
»
```

PSUB   Perform Polynomial Subtraction   (76)

28.5 bytes   #DAD9h

2: [polynomial 1]		2:
1: [polynomial 2]	====>	1: [poly 1 – poly 2]

```
« NEG PADD »
```

P→SYM   Convert a Polynomial to Symbolic Form   (76)

115 bytes   #30BEh

1: [polynomial]	====>	1: symbolic function

```
«  -3 CF
   →  v
   «  OBJ→ OBJ→ DROP 0 SWAP 1
      FOR n n 1 + ROLL v n 1 - ^ * + -1 STEP
      10 FIX →Q STD
   »
»
```

PTAYL   Plot a Taylor Series Approximation   (45)

343 bytes   #B29Ch

5: Taylor series approximation		5:
4: function being approximated		4:
3: independent variable		3:
2: point around which approx. is centered		2:
1: point being approximated	====>	1:

```
«  -3 CF
   →  p f v a b
   «  { PPAR PICT } PURGE f p 2 →LIST STEQ FUNCTION RAD v
      INDEP a DUP DUP 2 * b 2 * - ABS 6 MAX DUP ROT SWAP -
      IF DUP v STO f EVAL TYPE 0 ≠ THEN DROP 0 END
      3 ROLLD + DUP2 2 →LIST 3 ROLLD XRNG 1
      «  v STO f EVAL IF DUP TYPE 1 == THEN DROP 0 END »
      DOLIST OBJ→ DROP 6 + SWAP 6 - -.5 MIN SWAP YRNG a v
      STO { 1 1 } ATICK ERASE DRAX DRAW PICTURE
   »
»
```

→Qc   Convert a Decimal to a Simple Fraction (If possible)   (198)

417 bytes   #679Bh

1: decimal	====>	1: simple fraction or decimal

```
«  -55 CF DUP SIGN OVER ABS 10 { } (1,0) (0,1)
   →  z sign x n k r s
```

```
    «  1 n
       START
          x ABS FLOOR 'k' OVER STO+ x ABS OVER -
          IFERR INV THEN DROP 9.99999999999E499 END
          'x' STO r * s + r 's' STO 'r' STO r
       NEXT
       n →LIST k 1 ADD 2
       «  IF / .05 < THEN 1 ELSE 0 END » DOSUBS 1 POS
       IF DUP
       THEN GET C→R "'" ROT + "/" + SWAP + OBJ→ sign *
       ELSE DROP2 z
       END
    »
»
```

RELRT **Solve a Related Rates Problem** **(115)**

1541 bytes # 6D2h

3: Relation or list of relations	3:
2: List of variables	2:
1: Solution variable or list of solution variables ====>	1: Solution or list of solutions

```
«  0 'MARKER' STO { -3 -55 1 } CF 3 ROLLD DUP
   IF DUP 't' POS 0 == THEN 't' SWAP + END
   ROT
   IF DUP TYPE 5 ≠ THEN 1 →LIST END
   DUP ROT 1 1 4 PICK SIZE
   FOR j
      IF j 1 - THEN 1 + OVER SWAP END
   NEXT
   →LIST 2 « IMPδ » DOLIST
   → s v f d
   «  1 f SIZE
      FOR k
         v 1
         «  f k GET SWAP DUP DUP DUP "'" SWAP + "0" + OBJ→
            ROT "'" SWAP + "t" + OBJ→ DUP 4 ROLLD ROT "'δ"
            SWAP + ".t" + OBJ→ 3 →LIST 4 ROLLD 2 →LIST DUP
            3 ROLLD ↓MATCH DROP f k ROT PUT 'f' STO d k GET
            SWAP ↓MATCH DROP d k ROT PUT 'd' STO
         »
         DOLIST
      NEXT
      DUP 1 « OBJ→ DROP 't' * ROT + = » DOLIST
      f 1 « EVAL » DOLIST + d +
      →  vars eqns
      «  f SIZE 1 - DROPN
         WHILE
            IF 1 FC?C
            THEN
               "RELATED RATES" { "TIME(t):" "ENTER TIME WITH
               OR WITHOUT UNITS" 0 9 13 } { } { } vars 1
               «  1
                  «  "" + ":" + 1 →LIST "ENTER VALUE/UNITS,
                     IF KNOWN" + 0 + 9 + 13 +
                  »
                  DOLIST EVAL
               »
```

```
            DOLIST OBJ→ 3 + →LIST { 3 1 } { } { } INFORM
          ELSE 0
          END
       REPEAT
          → vals
          « CLLCD "Solving . . ." 3 DISP
            WHILE vals DUP { NOVAL } 1 GET POS DUP DUP
            REPEAT
               IF DUP 3 MOD 2 ==
               THEN
                  3 / CEIL DUP eqns SWAP GET OBJ→ DROP2
                  OBJ→ 3 DROPN = eqns 3 ROLLD PUT 'eqns'
                  STO
               ELSE DROP
               END
               0 PUT 'vals' STO
            END
            DROP2 vars 1 « OBJ→ DROP » DOLIST 't'
            SWAP + DUP 'varx' STO STO eqns STEQ MINIT
            vals 1
            « IF DUP UVAL
               THEN vals SWAP POS varx SWAP GET
               ELSE DROP
               END
            »
            DOLIST MUSER "ALL"
            IFERR MROOT
            THEN
               CLLCD
               IF ERRN # A01h ==
               THEN "Incompatible or missing units"
               ELSE DROP ERRM
               END
               MSGBOX "Try again . . ." 3 DISP 1 CF
            ELSE
               IF s DUP TYPE 5 ≠ THEN 1 →LIST END
               1 « DUP RCL SWAP →TAG » DOLIST EVAL 1 SF
            END
          »
       END
    »
    VARS DUP 'MARKER' POS 1 SWAP SUB PURGE
  »
»
```

REMNDR Convert a Quotient and Remainder to Symbolic Form (76)

276 bytes #EE73h

4:		4: [Quotient]
3: [Quotient]		3: [Numerator of remainder]
2: [Numerator of remainder]		2: [Denominator of remainder]
1: [Denominator of remainder]	====>	1: 'Symbolic expression'

```
« -3 CF
  → q n d
  « IF n [ 0 ] ≠
    THEN
       q OBJ→ 1 GET
```

```
          → t k
          «  0 k →ARRY t 1 →ARRY d PMULT n PADD 'x' P→SYM d
             'x' P→SYM / SWAP 'x' P→SYM SWAP +
          »
        ELSE q 'x' P→SYM
        END
        q n d 4 ROLL
     »
  »
```

RF→S Convert a Rational Fraction to a Symbolic Expression (76)

67 bytes #EB51h

2: [Numerator polynomial]		2:
1: [Denominator polynomial]	====>	1: 'Symbolic expression'

```
« 2 →LIST 1 « 'x' P→SYM » DOLIST EVAL / »
```

RROOTS Find the Real Roots of a Polynomial (310)

141 bytes #54ECh

1: polynomial (array)	====>	1: list of real roots

```
«  PROOT DUP SIZE 1 GET
   → r n
   «  { }
      1 n
      FOR k
         r k GET IM
         IF THEN r k GET + ELSE r k GET RE SWAP + END
      NEXT
   »
»
```

→RPN Convert an Algebraic Object to an RPN List (285)

189.5 bytes #6FB6h

1: 'algebraic'	====>	1: { list }

```
«  OBJ→
   IF OVER
   THEN
      → n f
      «  1 n
         FOR i
            IF DUP TYPE 9 SAME THEN →RPN END
            n ROLLD
         NEXT
         IF DUP TYPE 5 ≠ THEN 1 →LIST END
         IF n 1 >
         THEN
            2 n
```

```
            START + NEXT
         END
         f +
       »
    ELSE 1 →LIST SWAP DROP
    END
»
```

SCOF Find a Symbolic Cofactor (322)

254.5 bytes #EAE0h

3: symbolic matrix			
2: row		2:	
1: column	====>	1:	cofactor

```
«  -3 CF 3 PICK DUP SIZE SWAP 1 GET SIZE DROP
   IF 1 ==
   THEN 3 DROPN 1
   ELSE
      → r c
      «  OBJ→ OVER SIZE OVER 1 -
         → m n
         «  r - 1 + ROLL DROP
            1 n
            START
               n ROLL
               IF c 1 -
               THEN DUP 1 c 1 - SUB SWAP c 1 + m SUB +
               ELSE 2 m SUB
               END
            NEXT
            n →LIST
         »
      »
      SDET
   END
»
```

SCROSS Find the Cross Product of Two Symbolic Vectors (225)

251 bytes #724Ah

2: Symbolic vector 1		2:	
1: Symbolic vector 2	====>	1:	SV1 X SV2

```
«  -3 CF
   →  v w
   «  v w 2 →LIST 1
      «  IF DUP SIZE 2 == THEN 0 + END
      »
      DOLIST { v w } STO v 2 3 SUB w 2 3 SUB REVLIST *
      EVAL - v HEAD w 3 GET * v 3 GET w HEAD * - NEG v 1 2
      SUB w 1 2 SUB REVLIST * EVAL - 3 →LIST COLCT
   »
»
```

SDET Find a Symbolic Determinant (290)

136.5 bytes #575Dh

1: symbolic matrix	====>	1: determinant

```
«  -3 CF DUP DUP SIZE SWAP 1 GET SIZE DROP
   → a n
   «  0 1 n
      FOR i a i GET 1 GET a i 1 SCOF * -1 i 1 + ^ * +
      NEXT
   »
»
```

SDOT Compute the Dot Product of Two Symbolic Vectors (224)

57 bytes #9BE0h

2: Symbolic vector 1		2:
1: Symbolic vector 2	====>	1: SV1 • SV2

```
« -3 CF → m n « m n * ΣLIST » »
```

SEGINT Segment an Integral at a Given Value (160)

297.5 bytes #2F81h

2: Integral expression		2:
1: Value of point of segmentation	====>	1: Segmented integral

```
«  →  limit
   «  { '∫(&1,&2,&3,&4)' '∫(QUOTE(&1),limit,QUOTE(&3),&4)+∫(li
      mit,QUOTE(&2),QUOTE(&3),&4)' } ↓MATCH DROP EVAL {
      '∫(&1,&2,QUOTE(&3),&4)' '∫(&1,&2,&3,&4)' } ↓MATCH DROP
   »
»
```

→SEQ Compute Sequence Terms from a Recursive Definition (12)

350 bytes #C22Dh

3: recursive sequence definition		3:
2: list of initial values		2:
1: index number	====>	1: list of sequence terms

```
«  -3 CF '↑↓sqnc' CRDIR ↑↓sqnc 0 'MARKER' STO SWAP DUP SIZE
   →  f n a r
   «  a r n ABS
      FOR i
         1 r
         FOR j a j GET "a" j + OBJ→ STO NEXT
         f EVAL + DUP i DUP 2 + r - SWAP 1 + SUB 'a'
         STO VARS DUP 'MARKER' POS 1 - 1 SWAP SUB PURGE
      NEXT
```

```
      REVLIST TAIL REVLIST
      IF n 0 ≥ THEN REVLIST HEAD END
      'MARKER' PURGE UPDIR '↑↓sqnc' PURGE
   »
»
```

SERIES Find an Approximation of a Series to a Given Precision (36)

220 bytes #87A3h

4: defining function of series		4:
3: index variable		3:
2: beginning index value		2: computed approximation
1: number of decimal places desired	====>	1: last index value used

```
«  -3 CF
   →  f v j p
   «  STD 0
      DO DUP j v STO f EVAL + j 1 + 'j' STO DUP 1 DISP
      UNTIL
         IF p 3 + NEG DUP -11 ≥ THEN RND ELSE DROP END
         SWAP OVER == KEY OR
      END
      p
      IF DUP 11 ≤ THEN NEG END
      3 - RND p FIX j
   »
»
```

SERX1 Perform the Root Test for Convergence on a Series (32)

389.5 bytes #F6A8h

2: series expression		2:
1: index variable	====>	1: "Diverges", "Converges", or "Inconclusive"

```
«  →  f v
   «  f ABS v XROOT 0 1
      →  fp j d
      «  0
         DO
            j 1 + DUP 'j' STO ALOG →NUM v STO
            fp →NUM DUP ROT
         UNTIL
            DUP2 == 3 ROLLD DUP 'p' STO OVER - ABS SWAP
            / DUP d > SWAP 'd' STO OR
         END
         DROP d p v PURGE 'p' PURGE
      »
      IF SWAP ABS 9.99999999999E499 ==
      THEN DROP "Inconclusive"
      ELSE
         10 RND
         IF DUP 1 ≥
         THEN
            IF 1 == THEN "Inconclusive" ELSE "Diverges" END
```

```
            ELSE DROP "Converges"
            END
         END
      »
   »
```

SERX2 Perform the Limit Comparison Test on a Series (32)

169.5 bytes #F5A0h

3: target series expression		3:	
2: comparison series expression		2:	
1: index variable	====>	1: list of values toward limit	

```
«  3 ROLLD / 0 1
   →  v f j d
   «  0 0
      DO
         SWAP j 1 + DUP 'j' STO ALOG →NUM v STO
         f →NUM DUP2 4 ROLL
      UNTIL == 10 j == OR
      END
      SWAP j →LIST 3 ROLLD DROP2 v PURGE
   »
»
```

SERX3 Perform the Integral Test for Convergence on a Series (32)

182.5 bytes #C0CBh

2: series expression		2:
1: index variable	====>	1: list of values toward limit

```
«  →  f v
   «  1 FIX { 1 20 80 200 } 1
      «  →NUM ALOG 'b' STO 1 b f v ∫
         IFERR →NUM THEN RE END
      »
      DOLIST 'b' v 'IERR' 3 →LIST PURGE
   »
»
```

SM→ Disassemble a Symbolic Array (304)

124 bytes #46AAh

mn + 2:		mn + 2:
…		… elements
3:		3:
2:		2: # of rows
1: symbolic array	====>	1: # of columns

```
«  OBJ→ OVER SIZE
   → row col
```

```
   « 1 row
     FOR i i 1 - col * row + i - 1 + ROLL OBJ→ DROP NEXT
     row col
   »
»
```

→SM Assemble a Symbolic Array (325)

106.5 bytes #EBA7h

mn + 2:		mn + 2:	
… elements		…	
3:		3:	
2: # of rows		2: # of rows	
1: # of columns	====>	1: # of columns	

```
« → row col
   « 1 row
     FOR i col →LIST col row i - * i + ROLLD NEXT
     row →LIST
   »
»
```

SOLVPLT Plot a Relation Using Solver to Compute One Variable (104)

593 bytes #608Eh

5: List of relations to plot		5:
4: List of variables: {indep depnd }		4:
3: Plotting interval: {begin end }		3:
2: List of starting guesses for depnd		2:
1: Plot resolution/speed factor	====>	1:

```
« DUP SIGN SWAP ABS R→B RES RCLF { -3 -31 -55 } CF
  IFERR RCEQ THEN 1 END
  « → eq dep
     « IFERR « eq dep DUP EVAL ROOT » STEQ DRAX DRAW
        THEN 1
        ELSE 0
        END
        IF THEN ←flags STOF ERRN DOERR END
     »
  »
  → f v rng vals s ←flags old ←plot
  « FUNCTION
     IF s 1 ==
     THEN
        ERASE v 1 GET rng + INDEP rng DUP 2 * EVAL XRNG
        EVAL YRNG
     ELSE v DUP PURGE EVAL DEPND INDEP
     END
     1 f SIZE
     FOR k
        f k GET v 2 GET vals k GET
        IF DUP TYPE 5 ==
        THEN
```

```
            DROP 1 vals SIZE
            FOR j DUP2 vals j GET v 2 GET STO ←plot EVAL
            NEXT DROP2
         ELSE v 2 GET STO ←plot EVAL
         END
      NEXT
      PICTURE ←flags STOF
      IF old DUP TYPE THEN STEQ ELSE DROP 'EQ' PURGE END
   »
»
```

SPLEVAL Evaluate the Cubic Spline at a Given Point (146)

512.5 bytes #BDD6h

4: List of independent coordinates		4:
3: List of dependent coordinates		3:
2: List of spline coefficients		2:
1: Value of independent variable to interpolate	====>	1: Interpolated value

```
«  OVER SIZE DUP 1 DUP2
   →  us vs ys x n hi lo k h
   «  WHILE hi lo - 1 >
      REPEAT
         hi lo + 2 / 'k' STO
         IF us k GET x >
         THEN k 'hi' STO
         ELSE k 'lo' STO
         END
      END
      us hi GET us lo GET - 'h' STO
      IF h
      THEN
         us hi GET x - h / x us lo GET - h /
         →  a b
         «  a vs lo GET * b vs hi GET * + a 3 ^ a - ys lo
            GET * b 3 ^ b - ys hi GET * h 2 ^ * + 6 / +
         »
      ELSE "Not a function" DOERR
      END
   »
»
```

SPLINE Compute Cubic Spline Coefficients for a Data Set (146)

864 bytes #9BAEh

4: List of ind. variable data		4:
3: List of dep. variable data		3: List of ind. variable data
2: Number of data points used		2: List of dep. variable data
1: List of range over which spline is computed	====>	1: List of spline coefficients

```
«  EVAL { -.5 } DUP
   →  us vs n yp1 ypn ys u
   «  3 us 1 2 SUB EVAL SWAP - DUP 3 ROLLD / vs
      1 2 SUB EVAL SWAP - ROT / yp1 - * u 1
```

```
        ROT PUT 'u' STO 2 n 1 -
        FOR i
           us i GET us i 1 - GET DUP us i 1 + GET SWAP -
           3 ROLLD - SWAP / DUP ys i 1 - GET * 2 + DUP2 SWAP
           1 - SWAP / ys SWAP + 'ys' STO 6 vs i 1 + GET vs
           i GET - us i 1 + GET us i GET - / * vs i GET vs
           i 1 - GET - us i GET us i 1 - GET - / - us i 1 +
           GET us i 1 - GET - / ROT u i 1 - GET * - SWAP /
           u SWAP + 'u' STO
        NEXT
        ys 3 us n DUP 1 - SWAP SUB EVAL SWAP - DUP 3
        ROLLD / ypn vs n DUP 1 - SWAP SUB EVAL SWAP - - ROT
        / * .5 u n 1 - GET * - .5 ys n 1 - GET * 1 + / +
        'ys' STO n 1 - 1
        FOR k
           ys k GET ys k 1 + GET * u k GET + ys k ROT PUT
           'ys' STO -1
        STEP
        us vs ys
     »
  »
```

SREVH Compute the Volume of a Solid of Revolution (203)

759 bytes #2184h

5:	{ lower ...other intersection pts. ... upper }		5:
4:	Boundary function 1		4:
3:	Boundary function 2		3:
2:	Variable of integration		2:
1:	*k* such that *y* = *k* is axis of revolution	====>	1: Volume of revolved solid

```
«  { -3 -55 } CF DUP DUP 6 ROLL DUP ROT - 3 ROLLD 3 ROLLD
   6 ROLL DUP ROT - ROT SWAP DUP2 DUP2 SIGN SWAP SIGN == 3
   ROLLD ABS SWAP ABS ≤ DUP NOT
   →  c v l r s rl sl m n q
   «  { 'ABS(sl)^2' 'ABS(sl)^2-ABS(rl)^2' 'ABS(rl)^2'
      'ABS(rl)^2-ABS(sl)^2' } π *
      →  f
      «  c 2
         «  →  a b
            «  { 'r-s' 'r-l' 's-l' 'r+s' } 1
               «  v a b 2 →LIST
                  IFERR ROOT THEN 9.99999999999E499 END
                  IF DUP DUP a > SWAP b < AND NOT
                  THEN DROP
                  END
               »
               DOLIST
               IF DUP TYPE 5 ==
               THEN
                  SORT 1 « RCLF SWAP 8 FIX →Qπ SWAP STOF »
                  DOLIST
               ELSE { }
               END
               a SWAP + b + 2 « DUP2 + 2 / v STO f m
               n 2 * + 1 + →NUM GET v DUP PURGE SHOW v ∫ »
               DOSUBS
```

```
            IF DUP SIZE 1 > THEN ΣLIST ELSE 1 GET END
            →NUM
          »
        »
        DOSUBS
        IF DUP SIZE 1 > THEN ΣLIST ELSE 1 GET END
      »
    »
  »
```

S→RF Convert a One-Variable Algebraic to a Polynomial Ratio (76)

2004.5 bytes #C86Bh

2:		2: [numerator polynomial]
1: 'Algebraic'	====>	1: [denominator polynomial]

```
« -3 CF DEPTH SWAP
  IF DUP DUP TYPE 9 == THEN →RPN ELSE 1 →LIST END
  DUP SIZE
  « → a b
    « a b 2 →LIST 1 2
      FOR j
        IF DUP j GET TYPE 5 == THEN j ELSE 0 END
        j SWAP PUT
      NEXT
      ΣLIST
      → t
      « { [ 1 ] [ 1 ] } { d1 d2 } STO
        IF t
        THEN
          IF t 1 ==
          THEN a { n1 d1 } STO b 'n2' STO
          ELSE
            IF t 2 ==
            THEN b { n2 d2 } STO a 'n1' STO
            ELSE a { n1 d1 } STO b { n2 d2 } STO
            END
          END
        ELSE a 'n1' STO b 'n2' STO
        END
        t
      »
    »
  »
  « IF DUP2 ←case EVAL
    THEN
      DROP2 n1 d2 PMULT n2 d1 PMULT PADD d1 d2 PMULT 2
      →LIST
    ELSE PADD
    END
  »
  « IF DUP2 ←case EVAL
    THEN
      DROP2 n1 d2 PMULT n2 d1 PMULT PSUB d1 d2 PMULT 2
      →LIST
    ELSE PSUB
    END
  »
```

```
«  IF DUP2 ←case EVAL
   THEN DROP2 n1 n2 PMULT d1 d2 PMULT 2 →LIST
   ELSE PMULT
   END
»
«  IF DUP TYPE 5 ==
   THEN DUP 1 GET NEG 1 SWAP PUT
   ELSE NEG
   END
»
«  IF ←case EVAL DUP
   THEN
      IF 2 <
      THEN
         n1 n2 1 GET ABS 1 →ARRY DUP 3 ROLLD PPOWER d1
         ROT PPOWER
         IF n2 1 GET SIGN 1 + NOT THEN SWAP END
         2 →LIST
      ELSE
         DEPTH 1 + ←d - DROPN ←f { n1 n2 d1 d2 } PURGE
         "Non-integer exponent" DOERR
      END
   ELSE
      DROP
      IF n2 1 GET DUP 0 <
      THEN [ 1 ] n1 ROT ABS 1 →ARRY PPOWER 2 →LIST
      ELSE DROP n1 n2 PPOWER
      END
   END
»
«  IF DUP2 ←case EVAL
   THEN DROP2 n1 d2 PMULT d1 n2 PMULT 2 →LIST
   ELSE 2 →LIST
   END
»
→  ←d ←f ←p n ←case ←plus ←minus ←mult ←neg ←pow ←div
«  1 n
   FOR k
      ←p k GET
      IF DUP TYPE
      THEN
         IF DUP TYPE { 6 7 } SWAP POS
         THEN DROP [ 1 0 ] 1 →LIST
         ELSE
            { + - * ^ NEG / } SWAP POS
            IF DUP
            THEN
               { « ←plus EVAL » « ←minus EVAL » «
               ←mult EVAL » « ←pow EVAL » « ←neg EVAL »
               « ←div EVAL » } SWAP GET 1 →LIST
            ELSE
               DEPTH 1 + ←d - DROPN ←f { n1 n2 d1 d2 }
               PURGE "Non-algebraic function" DOERR
            END
         END
      ELSE 1 →ARRY 1 →LIST
      END
      ←p k ROT REPL '←p' STO
   NEXT
   ←p EVAL EVAL
   IF DUP2 PDIV DROP [ 0 ] SAME
```

```
        THEN 3 ROLLD DROP2 [ 1 ]
        ELSE DROP
        END
        { n2 n1 d2 d1 } PURGE
    »
»
```

SRFINT Find a Surface Integral (278)

530.5 bytes #7A8Ch

5: Multivariable function		5:
4: Parametrized surface (symbolic vector)		4:
3: List of function variables		3:
2: List with first parameter and its range		2:
1: List with second parameter and its range	====>	1: Surface integral

```
«  -3 CF DUP2 HEAD PURGE HEAD PURGE
   → f s x u v
   «  v TAIL EVAL u TAIL EVAL s u HEAD VDER s v HEAD VDER
      SCROSS COLCT s x STO
      IF f DUP TYPE 9 ==
      THEN EVAL COLCT SWAP VABS COLCT * COLCT
      ELSE 1 « EVAL COLCT » DOLIST SDOT COLCT
      END
      u HEAD ∫ EVAL COLCT
      IF DUP TYPE
      THEN
         IF OBJ→ DUP { ∫ } SWAP POS
         THEN
            SWAP DROP EVAL HALT EVAL COLCT
            IF DUP TYPE
            THEN
               IF OBJ→ DUP { ∫ } SWAP POS
               THEN SWAP DROP EVAL v HEAD ∫ MLTINT
               ELSE SWAP DROP EVAL v HEAD ∫ →NUM
               END
            ELSE v HEAD ∫ →NUM
            END
         ELSE SWAP DROP EVAL v HEAD ∫ →NUM
         END
      ELSE v HEAD ∫ →NUM
      END
      x PURGE STD
   »
»
```

TNFCN Find the Tangent and Normal to a Function at a Point (102)

237.5 bytes #6A53h

3: function		3:
2: independent variable		2: Normal line
1: Value of indep. at pt. of tangency	====>	1: Tangent line

```
«  → f v x
   «  x v STO f v ∂ DUP
```

```
      IF DUP 0 ==
      THEN DROP2 'y' f EVAL = v x =
      ELSE
         INV NEG 2 →LIST 1
         « 'y' SWAP DUP v * f →NUM ROT x * - + =
         » DOLIST EVAL
      END
      "Norm" →TAG SWAP "Tang" →TAG v PURGE
   »
»
```

TPLAN Find a Plane Tangent to a Surface at a Point (243)

148.5 bytes #D01Fh

3: Function		3:	
2: List of variables		2:	
1: List of coordinates of point	====>	1:	Equation of tangent plane

```
«  → f v p
   « v DUP PURGE p - COLCT f DUP v GRADI p v STO
     1 « →NUM » DOLIST SWAP EVAL SWAP ROT v PURGE
     SDOT + EXCO
   »
»
```

TYLRa Compute a Taylor Series About $x = a$ (44)

145.5 bytes #7410h

4: function being approximated		4:	
3: independent variable		3:	
2: order of Taylor series desired		2:	
1: point about which series is computed	====>	1:	Taylor series

```
«  -3 CF
   → f v o a
   « v PGALL v 'a' + v STO f EVAL v o TAYLR v DUP PURGE
     'a' - v STO EVAL v PURGE EVAL
   »
»
```

TYLRERR Estimate the Error in a Taylor Approximation (53)

94.5 bytes #70C6h

4: estimated max. for next derivative		4:	
3: order of approximation		3:	
2: point about which approx. is centered		2:	
1: point being approximated	====>	1:	estimated error

```
«  → d o a b
   « d o 1 + ! / b a - o 1 + ^ * ABS
   »
»
```

*** *Be sure to read the instructions on pages 283-284 before keying in these programs.* ***

UBINT Find an Improper Integral Over an Unbounded Interval (165)

216.5 bytes #7097h

1: Integral expression (including infinity) ====>	1: Evaluated integral

```
«  MAXR '∞' STO DUP OBJ→ DROP2
   IF DUP2 INDEF
   THEN
      DUP 5 PICK 4 PICK STO EVAL SWAP 6 PICK 4 PICK STO
      EVAL - COLCT →NUM SWAP PURGE 5 ROLLD 4 DROPN
   ELSE
      DROP 4 ROLLD 3 DROPN
      →  int v « int 'u' v ATAN = CHVAR →NUM v PURGE »
   END
   '∞' PURGE
»
```

UDER Find a Derivative Using Unit Objects (70)

94 bytes #BE9Eh

2: Function	2:
1: Variable of differentiation ====>	1: Derivative

```
«  PATH 3 ROLLD HOME '↓↓↑' CRDIR FDER UPDIR '↓↓↑' PGDIR SWAP
   EVAL EVAL
   IF DUP TYPE 9 == THEN COLCT END
»
```

VABS Find the Absolute Value of a Symbolic Vector (226)

78.5 bytes #455Dh

1: Symbolic vector ====>	1: Symbolic absolute value of vector

```
«  -3 CF
   →  v
   «  0 1 v SIZE
      FOR k v k GET 2 ^ + NEXT
      √
   »
»
```

VDER Find the Partial Derivative of a Symbolic Vector (82)

74.5 bytes #986Ch

2: Symbolic vector	2:
1: Variable of differentiation ====>	1: Partial derivative of vector

```
«  →  v x
   «  v 1 « x FDER COLCT » DOLIST
   »
»
```

VDIR Compute the Direction Angles of a Symbolic Vector (226)

139 bytes #8A73h

1: Vector (symbolic or numeric)	====>	1: List of direction angles

```
«  -3 CF OBJ→
   IF DUP TYPE 5 == THEN 1 GET END
   →LIST DUP SIZE SWAP DUP VABS
   → n v m
   «  1 n
      FOR i v i GET m / ACOS NEXT
      n →LIST
   »
»
```

VDIV Find the Divergence of a Vector Function (254)

80 bytes #2DE7h

2: Symbolic vector function		2:
1: List of variables	====>	1: Divergence

```
«  → f v
   «  f v 2 « FDER » DOLIST ΣLIST COLCT
   »
»
```

XINDEF Find an Indefinite Integral with an Expanded Search (180)

666.5 bytes #311Bh

2: Function		2: Indefinite integral
1: Variable of integration	====>	1: 1/0 (success or failure)

or

2:		2: Indefinite integral
1: Symbolic integral expression	====>	1: 1/0 (success or failure)

```
«  -3 CF
   IF DUP TYPE 9 ==
   THEN OBJ→ DROP2 ROT DROP ROT DROP
   END
   DUP
   →  f ↑↓ w
   «  IF f TYPE NOT THEN '0*x' f + 'f' STO END
      1 ↑↓ f ↑↓ SHOW '↑↓' DUP STO OVER XINT
      IF DUP OBJ→ DUP { ∫ } 1 GET SAME
      THEN DROP DROPN 0
      ELSE
         5 ROLL DROP
         IF { + } 1 GET SAME
         THEN
            →LIST 0 SWAP 1
```

```
            « IF DUP { '∫(&1,&2,&3,&4)' ↑↓ } ↓MATCH
              THEN
                 DROP COLCT EVAL
                 IF DUP { '∫(&1,&2,&3,&4)' ↑↓ } ↓MATCH
                 THEN DROP SWAP
                 ELSE
                    DROP OBJ→ 3 DROPN EVAL '↑↓' w 2 →LIST
                    ↓MATCH DROP SWAP .5 +
                 END
              ELSE
                 DROP OBJ→ 3 DROPN EVAL '↑↓' w 2 →LIST
                 ↓MATCH DROP SWAP .5 +
              END
            »
            DOLIST OBJ→ 1 + ROLLD + SWAP
        ELSE DROP2 EVAL '↑↓' w 2 →LIST ↓MATCH DROP 1
        END
     END
     IF DUP .5 == THEN .5 - END
  »
»
```

XINT Perform an Integration with an Expanded Pattern Search (175)

548 bytes #190Dh

4: lower bound		4:	
3: upper bound		3:	
2: integrand		2:	
1: variable of integration	====>	1: Evaluated integral, if possible	

```
« → l u i x
  « -3 CF
    « → low up int v
      « int v { ↑↓ } + ↑MATCH DROP IPATS DUP SIZE 0
        → pats n t
        « DO pats n GET
          UNTIL ↓MATCH 't' STO 'n' DECR NOT t OR
          END
          IF t
          THEN
             DUP up 2 →LIST 'f' APPLY SWAP low 2
             →LIST 'f' APPLY - { 'f(&1,&2)' '&1|(↑↓=&2)'
             } ↓MATCH DROP
          ELSE DROP low up int v ∫
          END
        »
      »
    »
    → f
    « l u i x ∫ { '∫(&1,&2,&3,&4)'
      'f(&1,&2,QUOTE(&3),QUOTE(&4))' }
      IF ↓MATCH THEN EVAL END
    »
  »
»
```

XPOW Perform Alternative Version of Y^X Command (335)

121 bytes #AC3Dh

2: Base	2:
1: Exponent ====>	1: Base^Exponent

```
«  → y x
   « IF x DUP IP ≠
     THEN
       x ABS →Qc OBJ→ DROP2 y SWAP XROOT SWAP x SIGN
       * ^
     ELSE y x ^
     END
   »
»
```

XRCONV Convert a Symbolic Expression Containing *n*th-Roots (287)

96 bytes #954Dh

1: Symbolic expression ====>	1: Converted expression

```
«  →RPN 1
   « IF DUP { ^ } 1 GET SAME THEN DROP 'XPOW' END »
   DOLIST EVAL
»
```

YCOMP Create a Composite Plot of Y-Slices (231)

549.5 bytes #1883h

6: Function of two variables	6:
5: List of variables	5:
4: List of x-range	4:
3: List of y-range	3:
2: List of z-range	2:
1: Number of slices ====>	1:

```
«  →  f v xr yr z n
   «  f STEQ v DUP PURGE EVAL DEPND INDEP xr EVAL XRNG yr
      EVAL YRNG FUNCTION ERASE z EVAL - ABS n 1 - /
      → d
      « 1 n 2 -
        FOR k z HEAD d k * + NEXT
        n 2 - →LIST z EVAL ROT SWAP + + 1 « v 2 GET
        DUP ROT SWAP STO "y=" SWAP EVAL + 1 →GROB # 64h #
        5h BLANK PICT { # 1h # 1h } ROT REPL PICT { # 1h
        # 1h } ROT GOR DRAW » DOLIST # 64h # 5h BLANK PICT
        { # 1h # 1h } ROT REPL "y={" z EVAL →STR " }" +
        SWAP " " + SWAP + + 1 →GROB PICT { # 1h # 1h }
        ROT GOR PICTURE v PURGE
      »
   »
»
```

I. INDEXES

Examples Index

1. Series, Sequences, and Limits

2. Differentiation and the Derivative

3. Applications of the Derivative

4. Integration and the Integral

5. Applications of the Integral

6. Multivariate and Vector Calculus

Programs Index

Subject Index

If you liked this book, there are others that you will certainly enjoy also:

An Easy Course in Using and Programming the HP 48G/GX

Here is an *Easy Course* in true Grapevine style: Examples, illustrations, and clear, simple explanations give you a real feel for the machine and how its many features work together. First you get lessons on using the Stack, the keyboard, and on how to build, combine and store the many kinds of data objects. Then you learn about programming—looping, branching, testing, etc.—and you learn how to customize your directories and menus for convenient "automated" use. And the final chapter shows example programs—all documented with comments and tips.

Algebra and Pre-Calculus on the HP 48G/GX

Get ready for math class! Plot and solve with this great collection of lessons, examples and programs from an experienced math teacher:

Functions; trigonometry (identities, triangle solutions, other applications); polar and parametric equations, complex numbers; polynomials (graphs and roots); linear systems (matrix math, simultaneous equations, determinants, inequalities, linear programming); analytic geometry (lines, planes, translations, rotations, reflections); conic sections (circles parabolas, hyperbolas, ellipses).

Graphics on the HP 48G/GX

Here's a must-have if you want to use the full potential of that big display. Written by HP engineer Ray Depew, this book shows you how to build graphics objects ("grobs") and use them to customize displays with diagrams, pictures, 3D plots—even animation.

First the book offers a great in-depth review of the built-in graphics tools. Then you learn how to build your own grobs and use them in programs—with very impressive results!

The HP 48G/GX Pocket Guide

You get some 90 pages of quick-reference tables, diagrams, and handy examples—all in a convenient little booklet that ***fits in the case with your HP 48G/GX!*** There is a complete command reference, along with system flags, menus, application summaries, troubleshooting, and common Q's & A's. Nothing is more succinct and helpful than this little memory-jogger!

For more details on these books or any of our titles, check with your local bookseller or calculator/computer dealer. Or, for a full Grapevine catalog, write, call or fax:

ISBN		Price*
	Books for personal computers	
0-931011-28-0	**Lotus** Be Brief	$ 9.95
0-931011-29-9	A Little **DOS** Will Do You	9.95
0-931011-32-9	Concise and **WordPerfect**	9.95
0-931011-37-X	An Easy Course in Using **WordPerfect**	19.95
0-931011-38-8	An Easy Course in Using **LOTUS 1-2-3**	19.95
0-931011-40-X	An Easy Course in Using **DOS**	19.95
	Books for Hewlett-Packard Scientific Calculators	
0-931011-18-3	An Easy Course in Using the **HP-28S**	9.95
0-931011-25-6	**HP-28S** Software Power Tools: **Electrical Circuits**	9.95
0-931011-26-4	An Easy Course in Using the **HP-42S**	19.95
0-931011-27-2	**HP-28S** Software Power Tools: **Utilities**	9.95
0-931011-33-7	**HP 48S/SX** Graphics	19.95
0-931011-XX-0	**HP 48S/SX** Machine Language	19.95
0-931011-41-8	An Easy Course in Using and Programming the **HP 48G/GX**	19.95
0-931011-42-6	Graphics on the **HP 48G/GX**	19.95
0-931011-43-4	Algebra and Pre-Calculus on the **HP 48G/GX**	19.95
0-931011-44-2	Calculus on the **HP 48G/GX**	19.95
0-931011-45-0	The **HP 48G/GX Pocket Guide** *(Available 1/96)*	9.95
	Books for Hewlett-Packard Business calculators	
0-931011-03-5	An Easy Course in Using the **HP-12C**	19.95
0-931011-12-4	The **HP-12C Pocket Guide:** Just In Case	6.95
0-931011-19-1	An Easy Course in Using the **HP 19BII**	19.95
0-931011-20-5	An Easy Course In Using the **HP 17BII**	19.95
0-931011-22-1	The **HP-19B Pocket Guide:** Just In Case	6.95
0-931011-23-X	The **HP-17B Pocket Guide:** Just In Case	6.95
0-931011-XX-0	**Business Solutions** on Your HP Financial Calculator	9.95
	Books for Hewlett-Packard computers	
0-931011-34-5	**Lotus in Minutes** on the **HP 95LX**	9.95
0-931011-35-3	**The Answers You Need** for the **HP 95LX**	9.95
	Other books	
0-931011-14-0	**Problem-Solving Situations:** A Teacher's Resource Book	9.95
0-931011-39-6	**House-Training Your VCR:** A Help Manual for Humans	9.95

Contact: **Grapevine Publications, Inc.**
626 N.W. 4th Street P.O. Box 2449 Corvallis, Oregon 97339-2449 U.S.A.
800-338-4331 (503-754-0583) *Fax:* 503-754-6508

**Prices shown are as of 8/6/95 and are subject to change without notice. Check with your local bookseller or electronics/computer dealer—or contact Grapevine Publications, Inc.*

Reader Comments

We here at Grapevine like to hear feedback about our books. It helps us produce books tailored to your needs. If you have any specific comments or advice for our authors after reading this book, we'd appreciate hearing from you!

Which of our books do you have?

Comments, Advice and Suggestions:

May we use your comments as testimonials?

Your Name: Profession:

City, State:

How long have you had your calculator?

Please send Grapevine catalogs to these persons:

Name ______________________________________

Address ____________________________________

City ______________ State __________ Zip ___________

Name ______________________________________

Address ____________________________________

City ______________ State __________ Zip ___________

Calculus on the HP 48G/GX

Grab your calculator and this book and get ready now for your calculus class! You'll get lots of lessons, examples and advice on graphing and problem-solving throughout a three-term sequence of college-level calculus.

The book begins with a chapter on ***limits, sums and series***, including convergence tests. The next two chapters cover single-variable **differentiation** and **derivatives** (including formal, step-wise, and implicit methods), and their many **applications** in real-world problems, such as related rates and optimization .

Chapter 4 is all about **single-variable integration**—numeric and symbolic—including integration by parts, *u*-substitution and improper integrals. Chapter 5 illustrates the use of **integration in real-world problems**, such as averages, surfaces and volumes.

Finally there's a chapter on ***multi-variate and vector calculus***, including partial derivatives, gradients, path and surface integrals, and surfaces of revolution.

You get all this from an experienced classroom math teacher, plus a huge collection of programs to automate much tedious keystroking. Don't miss this valuable aid for your calculus courses!

ISBN 0-931011-44-2

51995

9 700931 011441

Grapevine Publications, Inc.

626 N.W. 4th St. P.O. Box 2449

Corvallis, OR 97339 U.S.A.