

COMPUTER CORNER

Edited by
Eugene A. Herman

In this column, readers are encouraged to share their expertise and experiences with computers as they relate to college level mathematics. Articles that illustrate how computers can be used to enhance pedagogy, solve problems, and model real-life situations are especially welcome.

All manuscripts for this column should be prepared according to the guidelines on the inside front cover and sent to:

Eugene A. Herman
Department of Mathematics
Grinnell College
Grinnell, IA 50112

Graphing with the HP-28S

John Selden
Annie Selden



John Selden received his Ph.D. in mathematics from the University of Georgia in 1963 and subsequently taught at various universities in the U.S. and abroad. He was Dean of Science at Bayero University in Kano, Nigeria before returning to the U.S. in 1985 to teach at Tennessee Technological University. His research interests are in topological semigroups, artificial intelligence, and tertiary mathematics education. He edits (with his wife, Annie Selden) the Research Sampler column of UME Trends. He is currently investigating inverse semigroups on the plane, analyzing misconcep-

tions in student proofs, and designing an experimental calculus sequence using notes and the HP-28S.



Annie Selden received her B.A. from Oberlin in 1959, her M.A. from Yale in 1962, and her Ph.D. from Clarkson in 1974. She has taught at universities in the U.S. and abroad. While at Bayero University in Kano, Nigeria, she administered the day-to-day affairs of the Mathematics Department. She has been teaching at Tennessee Technological University since 1985. Her research is in locally compact topological semigroups and tertiary mathematics education. She edits (with her husband, John Selden) the Research Sampler column of UME Trends. This paper arose from an

attempt to make the HP-28S more user-friendly for calculus students.

In teaching beginning calculus and other entry-level college courses, one would like to emphasize conceptual understanding and mathematical thinking, instead of routine manipulation. At one time this meant judiciously selecting examples in which the computations were not too difficult for students to carry out with pencil and paper. Now, using computers and calculators, it is technically possible for students to explore more complex examples on their own. Unfortunately, the time and tedium of calculation may simply be replaced by the time and tedium of learning to use a particular computing device.

One such device, the HP-28S, is now owned by many mathematics faculty and, if our university is typical, by quite a number of students. The HP-28S, with its 32K of memory, is really a pocket computer, and many observations on classroom use of microcomputers and computer algebra systems apply to it [1], [2], [3], [7]. This year we are teaching an experimental calculus sequence to volunteers, who are required to have this calculator. The aim is to find ways of teaching that enable students to solve nonroutine problems. In a recent study we found that average students from traditional calculus courses have great difficulty solving problems even slightly different from those presented in the text or classroom [6]. Our hope is that the HP-28S will enable the routine parts of calculus to be taught quickly [3], and, hence, provide time for students to practice solving novel problems. Our interest in the HP-28S is partly in its portability and availability to students at all times.

In exploring the capabilities of the HP-28S for this purpose, we discovered that students could easily get bogged down in learning how to use it efficiently. The flexibility needed in such a general-purpose scientific calculator prevents it from being maximally convenient for specialized applications, such as a beginning calculus course. Students could be taught to program the HP-28S to make it more convenient, but having students design appropriate systems of programs is time consuming and would interfere with the primary objective of our course. Commenting on the similar HP-28C, Tucker remarked, "It takes ten hours to become proficient enough to begin to realize the potential of this calculator, and one could spend weeks exploring the nooks and crannies of the machine" [8].

This paper presents a system of programs to make graphing easier on the HP-28S. They enable students to reflect on and refine a variety of examples without being distracted by the mechanics of operating the calculator. A typical function needs to be viewed in variously scaled rectangles or windows for greater accuracy [1] or to avoid coming to incorrect conclusions [2]. Sometimes one wants to zoom in on

a piece of the graph. Other times, one wants to look elsewhere for the interesting behavior, but a new window may give a worse view. OLWIN, one of the programs described here, recovers the previous window quickly. Some of the other programs work in ways similar to commands already available; e.g., Draw is similar to DRAW. The system that we are proposing can save time and frustration.

The graphing programs are called Draw, CLR, MOVE, ZOOM, WIN, and OLWIN and allow one to change the center and scale of the viewing window easily, as well as to recover the previous viewing window. Also included are two “housekeeping” programs, PARNT and Order, which should be installed before the graphing programs for efficient storage, as well as the program, DFix, which fixes a bug in the HP-28S command DRAW. DRAW works well in isolation but should not be called upon in a program; instead use DFix.

Perhaps the most useful of these programs is OLWIN because it saves time. In order to execute it, one needs Draw and DFix. The other programs presented in this paper can be omitted, if desired, although it is convenient to have them on the USER menu. A student can install the programs without understanding the details of how they work.

All existing commands on the PLOT menu, except DRAW, can be used in conjunction with these programs. Additionally, one can write other graphing programs (e.g., for polar coordinates) that are compatible. Compatibility can be assured by naming a new graphing program DFix and placing it in a subdirectory, say POLAR, below that of the programs given in this paper. Naming two programs DFix causes no problems as long as they are in two different directories. POLAR should also contain the program $\ll \{ \text{CLR Draw MOVE OLWIN WIN ZOOM PARNT} \} \text{ MENU} \gg$, which can be named Menu. When Menu is run, the listed programs (CLR, Draw, etc.) appear on the USER menu, and those calling on DFix will automatically substitute the new graphing program. This trick may seem odd, but it works well.

Two Scenarios

Consider the following effort of a naive student to view the graph of $y = x \sin \frac{1}{x}$ without aids or guidance as illustrated in Figure 1.

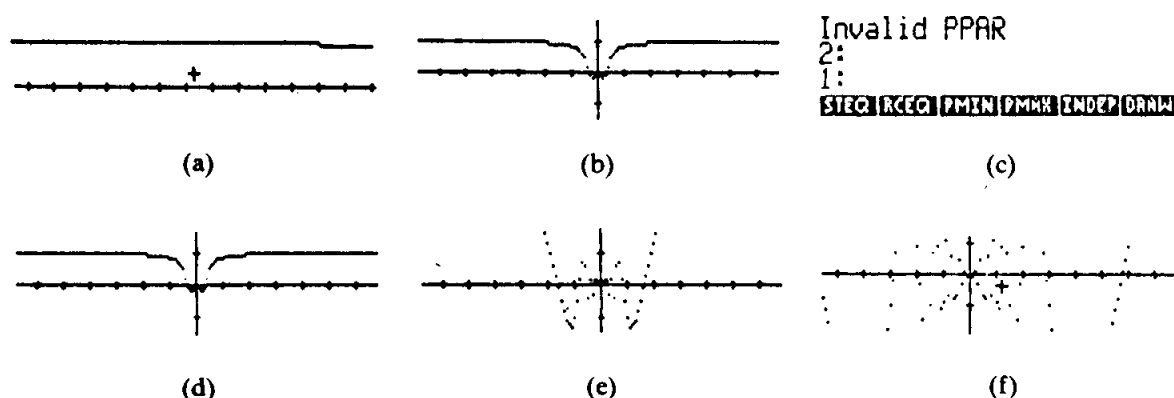


Figure 1. First Scenario: A naive student attempts to graph $x \sin \frac{1}{x}$ on the HP-28S.

Enter 'X*SIN(INV(X))' on the stack, store it in EQ using STEQ, found on the PLOT menu, and press DRAW. After some time, the result appears on the display, (a). Unfortunately, the plot parameters PPAR from a previous use of DRAW were not removed.

Enter 'PPAR' and PURGE. Press DRAW again, and after a while, (b) appears. Use the cursor keys and INS to choose the lower left and upper right corners of a window about the origin. Press PMIN, PMAX, and DRAW to get the display shown in (c)—not a graph, but an error message. PMIN and PMAX were executed in the wrong order, causing the window to have negative height and width.

Start again. Enter 'PPAR', PURGE, and DRAW and wait for (d). Next use the cursor keys and INS to enter the lower left and upper right corners of a new window. Then clear the display with ON and press PMAX, PMIN, and DRAW. Soon (e) appears. Do this again to get (f). It's clear such a student needs help.

Consider graphing the same function using the programs in this paper and the following advice. To graph a new function, always start with CLR to return the viewing rectangle to default position. If part of a graph looks like Figure 1(f), try using a square in WIN, thereby stretching the horizontal axis. To follow the behavior of a graph as it runs off the display, use MOVE. If a new display gives a worse view, use OLWIN to recover the original.

Enter the function $x \sin \frac{1}{x}$ as before, press CLR followed by Draw to get Figure 2(a). This suggests $y = 1$ is a horizontal asymptote. Next use 10 ZOOM to explore the graph near the origin, (b). As further use of ZOOM probably won't help, press OLWIN, which is very fast, to return to (a), and try WIN as follows.

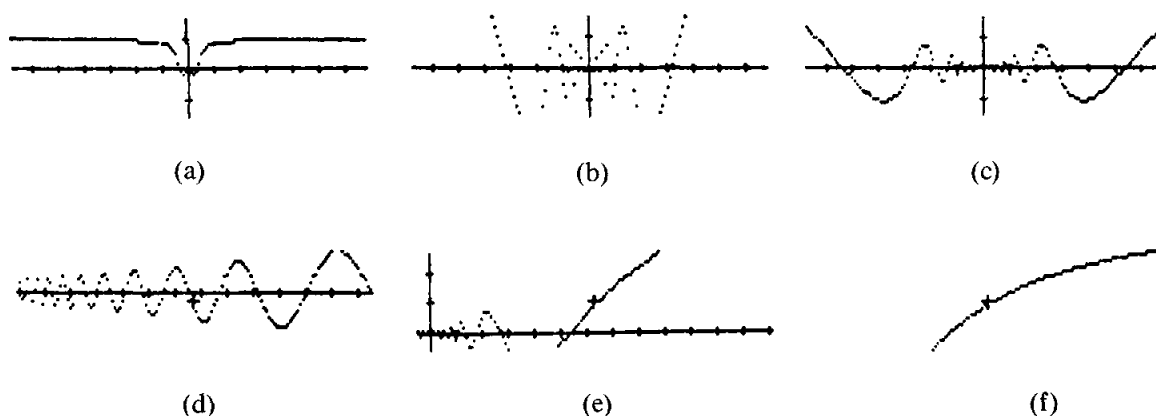


Figure 2. Second Scenario: A little advice and these programs give more information.

Use the cursor keys, followed by INS, to select a small square about the origin, press ON and WIN to get (c). Next use the cursor keys to get a window excluding the origin, and press WIN again to obtain a reasonably good local picture, (d). To follow the behavior of the graph to the right and see how (d) fits with the larger picture obtained in (a), press OLWIN to return to (c), move the cursor towards the upper right and press INS, ON, and MOVE to obtain (e). Do this again to get (f). This sequence gives a good overall idea of the graph.

A Few Technical Remarks

Although the HP-28S has output via a printer, the only way to input a program is to type it in. Thus, we have included not only descriptions of each of the programs, but also the actual code.

The programs may appear longer than necessary because safeguards have been included to prevent them from reacting badly when executed in unintended sequence or with unintended input. These safeguards are needed in the HP-28S, as variables can exist independently of the programs using them.

To install a program, enter it on the stack; i.e., type it in and press ENTER. Then enter the name of the program in single quotes, e.g., 'Draw', and press STO. DRAW will appear on the USER menu, which shows only capital letters. Commands need not be typed in; they may be entered from the menus. For example, PATH is on the MEMORY menu; see the alphabetical listing [5, beginning p. 324]. However, commands containing special symbols (e.g., \rightarrow) must be entered from the menus.

Running programs are indicated by the "busy" annunciator symbol, ((·)), on the display. Pressing ON during execution will normally stop a program but may leave unwanted entries on the stack. Programs that cannot be stopped in this way have entered an endless loop and can usually be stopped by performing a system halt, leaving the memory intact [4, p. 217]. The PRINT menu is inaccessible while a graph is displayed; however, the graph can be printed by pressing the ON and L keys simultaneously [4, p. 91].

Finally, the comments in italics are not part of the programs to be entered.

The Housekeeping Programs

PARNT. This little program changes the current directory from a subdirectory to its parent directory [4, p. 60]. It is often useful to group programs in directories. For example, these graphing programs might be put in a directory called GRAPH. Since the calculator can only find variables in the current directory or directories above the current one, it is necessary to be able to move around the directory tree. PARNT can be put in the top directory, called HOME, or copied into several subdirectories.

<<	<i>Start the program.</i>
{HOME}	<i>Put this one word list on the stack.</i>
PATH	<i>Enter the list of directories from HOME to the current one.</i>
+	<i>Combine the two lists into one.</i>
DUP	<i>Enter another copy of the list on the stack.</i>
SIZE	<i>Replace the list with its size, say n.</i>
1	<i>Put 1 on the stack.</i>
—	<i>Replace level two and level one with their difference, i.e., $n - 1$, which cannot be 0.</i>
GET	<i>Replace the path list and $n - 1$ with the $(n - 1)$-st entry in the list.</i>
EVAL	<i>Go to that directory, i.e., the parent directory.</i>
>>	<i>End the program.</i>

Order. This program arranges the presentation of program names on the USER menu to be: CLR, Draw, MOVE, OLWIN, WIN, ZOOM, Order. Without it, the names will move to the right as new variables are added on the left. Thus, the placement of the programs on the USER menu is constantly changing, and this may encourage user errors. Perhaps in the next version of the calculator this

unfortunate design characteristic will be altered. To ensure that Order executes properly, all the above programs must be in the current directory.

```

<<                                Start the program.
{CLR Draw MOVE                    Enter the list on the stack.
  OLWIN WIN
  ZOOM Order}
ORDER                             Order the directory according to the list.
>>                               End the program.

```

A Program to Fix DRAW

DFix. This program does what DRAW is supposed to do when it is called in a program; i.e., it draws the graph of the function(s) stored in EQ using the information in PPAR, the plot parameter, without leaving an unwanted number on the stack. Note that when DRAW or DFix is called in a program, neither automatically clears the display nor activates the cursor keys, unlike when DRAW is executed from the PLOT menu. The difficulty with calling DRAW in a program is that for certain functions, such as $x + \frac{1}{x}$, numbers are added to the stack because of an incorrectly designed loop. DFix should be put in the top directory, HOME.

```

<<                                Start the program.
RAD                               Convert to radian mode.
[1]                               Enter [1] to mark the end of the existing stack.
IFERR                             While executing DRAW, look for an error,
  DRAW                           perhaps from the nonexistence of EQ or PPAR.
THEN                             If there is a DRAW error, drop [1] and call
  DROP                           DRAW again to get the error message and
  DRAW                           abort programs containing DFix.
ELSE                             If there is no DRAW error, place
  [1]                             an additional [1] on the stack.
  WHILE                           While the last two entries are different, i.e.,
    SAME                           there is a number between the two [1]'s, add [1]
    NOT                             to the stack and repeat the test. Note that
  REPEAT                           executing SAME removes two entries from the
    [1]                             stack, e.g., the [1] and the number before it.
  END                               End WHILE.
END                               End IFERR.
>>                               End the program.

```

The Graphing Programs

Draw. This program appears to produce the same result as the command DRAW; i.e., it clears the display, draws the graph, and activates the cursor keys. In addition, Draw alters a variable, DVar, containing information necessary to retrieve the previous display.

When Draw is executed, it stores the current display in DVar as a string. DVar is a list of four objects: old and new versions of both PPAR and the display. Draw first replaces the old versions of PPAR and the display with the new versions. It then stores the current PPAR and display in DVar as the

new versions. If DVar does not exist, Draw creates it using the current PPAR and display for both old and new versions. Draw requires a function in EQ. This is provided using STEQ on the PLOT menu.

```

<<          Begin the program.
CLLCD      Clear the liquid crystal display.
DFix       Draw the graph (and create PPAR if needed).
PPAR       Enter PPAR on the stack.
LCD →      Enter the display as a string on the stack.
DVar       Enter DVar on the stack. If it does not exist,
           the name 'DVar' will be entered.

IFERR      Look for LIST → error in case DVar
  LIST →   did not exist.
THEN       If DVar did not exist, remove
  DROP     'DVar' from the stack.
  2         Duplicate two levels on the stack,
  DUPN     i.e., PPAR and the display string.
  4         Convert 4 levels of the stack
  → LIST   to a list.
  'DVar'   Enter 'DVar' on the stack and
  STO      store the list in it.
  Order    Reorder the user menu (optional and slow).
ELSE       If DVar existed, it is now on four
  ROLL     levels of the stack with a 4 on level 1.
  4         Interchange the first two with the
  ROLL     second two of these levels.
  DROP     Drop two levels from the stack.
  DROP
  4         Interchange the remaining two
  ROLL     levels with the two above.
  4
  ROLL     Convert 4 levels of the stack
  4         to a list.
  → LIST   Enter the name 'DVar' on the stack.
  'DVar'   Store the list in DVar.
  STO      End IFERR.
END        Activate the cursor keys.
DGTIZ
>>        End the program.

```

CLR. This program clears the current PPAR, DVar, and X from memory and should be executed before graphing a new function. In case an independent variable other than X is used, that variable should also be purged. Use of CLR assures that the viewing rectangle will be restored to default position in the next execution of Draw. Note that DVar uses about 1K of memory.

The use of errors for type checking is an integral part of these programs. When an error is detected, the stack is returned to its original state before the error. To be sure this happens, flag 31 must be set, i.e., contain a 1. This is the default setting; however, in case it is altered, CLR will restore it.

```

<<          Start the program.
31          Enter 31 on the stack.
SF          Remove 31 from the stack and set flag 31.
{PPAR DVar X} Enter the list of variables to be removed.
PURGE      Remove them.
>>        End the program.

```

MOVE. This program appears to produce the same result as CENTR followed by DRAW; i.e., it moves the center of the viewing rectangle to a point previously entered on the stack, often by means of the cursor keys. In addition, it alters DVar. For example, to follow a graph move the cursor to a point where the graph leaves the display and press INS, ON, and MOVE. The graph will be drawn again with that point in the center of the new display.

```
<<                                Start the program.
CENTR                            Take a point from the stack and record it as the
                                center component of the list PPAR.
Draw                             Graph the function using the new PPAR.
>>                             End the program.
```

ZOOM. This program divides the scale of the display by a number previously entered on the stack, alters DVar, and redraws the graph. For example, entering 10 ZOOM causes marks on the axes to represent $\frac{1}{10}$ their previous length, thereby enlarging what previously appeared in the middle of the display by a factor of 10.

```
<<                                Start the program.
INV                             Replace level one of the stack
DUP                             by its reciprocal and duplicate it.
IFERR                           Look for an error.
  *W                             Change the scale of the width.
THEN                             If there was an error, drop the extra entry.
  DROP                           Return the stack to its
  INV                             original form and call *W again
  *W                             for the error message.
ELSE                             If there was no *W error, change
  *H                             the scale of the height.
  Draw                           Graph the function using the new PPAR.
END                             End IFERR.
>>                             End the program.
```

WIN. This program assumes that two points were previously entered on the stack, the first one representing the lower left corner and the second the upper right corner of a viewing rectangle. Often these points will be obtained from the display by manipulating the cursor keys and pressing INS. WIN will then redraw the graph and alter DVar accordingly.

```
<<                                Start the program.
DUP                             Duplicate stack entry 1.
IFERR                           Look for an error.
  PMAX                           Record the second point in PPAR.
THEN                             If there was a PMAX error, then
  DROP                           drop the extra entry and
  PMAX                           call PMAX again for the error message.
ELSE                             If there was no PMAX error
  SWAP                           interchange the two points.
  IFERR                           Look for an error.
  PMIN                           Record the first point in PPAR.
THEN                             If there was a PMIN error, then
  SWAP                           return the stack to its
  1300                           original order and give a warning
  .07                            beep of 1300 c/sec. for .07 sec.
BEEP
```


ERRM	<i>Display the PMIN error message</i>
1	<i>on line 1 and end the program.</i>
DISP	
ELSE	<i>If there was no PMIN error, then</i>
DROP	<i>drop the remaining point.</i>
Draw	<i>Graph the function using the new PPAR.</i>
END	<i>End inner (second) IFERR.</i>
END	<i>End outer (first) IFERR.</i>
>>	<i>End the program.</i>

OLWIN. This program displays the previous graph. It also interchanges the new and old sections of DVar and adjusts PPAR appropriately. Pressing OLWIN twice returns one to the current graph. If DVar does not exist, OLWIN calls Draw.

<<	<i>Start the program.</i>
DVar	<i>Enter DVar or 'DVar' if DVar does not exist.</i>
IFERR	<i>Check the existence of DVar.</i>
LIST →	<i>Put DVar on 4 stack levels and put 4 on level 1.</i>
THEN	<i>If there was a LIST → error, DVar does</i>
DROP	<i>not exist so drop the name 'DVar' and</i>
Draw	<i>run Draw.</i>
ELSE	<i>If there was no LIST → error, reverse</i>
ROLLD	<i>the new and old DVar information</i>
4	<i>now on the stack by rolling down</i>
ROLLD	<i>4 places twice.</i>
2	<i>Duplicate two stack positions.</i>
DUPN	
SWAP	<i>Interchange stack level 1 and 2.</i>
'PPAR'	<i>Place the name 'PPAR' on the stack.</i>
STO	<i>Store the contents of level 2 in PPAR.</i>
5	<i>Roll down 5 levels on the stack.</i>
ROLLD	
4	<i>Convert 4 levels of the stack to a list.</i>
→ LIST	
'DVar'	<i>Store the list (the old DVar with new</i>
STO	<i>and old sections reversed) as DVar.</i>
CLLCD	<i>Clear the (liquid crystal) display.</i>
→ LCD	<i>Place the string representation of the old</i>
	<i>display on the display.</i>
DGTIZ	<i>Activate the cursor keys.</i>
END	<i>End IFERR.</i>
>>	<i>End the program.</i>

Finally several views of a graph can be stored in memory and retrieved quickly. To store a view enter DVar, a variable name, say 'G1', and STO. To retrieve a view enter G1, 'DVar', STO, and OLWIN. Since DVar uses 1K of memory, these variables should be purged when no longer needed.

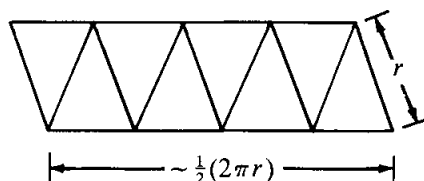
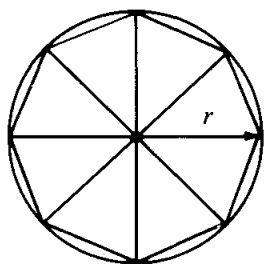
Acknowledgment. The authors thank Fred Linton of Wesleyan University for suggesting the improved version of PARNT included in this paper.

REFERENCES

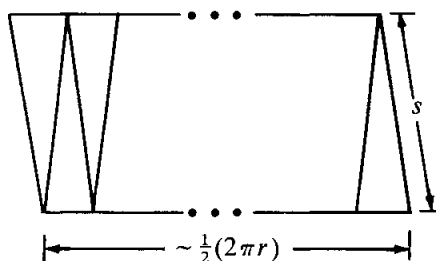
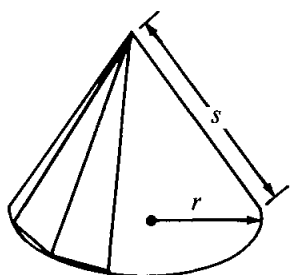
1. Franklin Demana and Bert K. Waits, Problem solving using microcomputers, *The College Mathematics Journal* 18 (May 1987) 236–241.

2. _____, Pitfalls in graphical computation, or why a single graph isn't enough, *The College Mathematics Journal* 19 (March 1988) 177-183.
3. M. Kathleen Heid, Resequencing skills and concepts in applied calculus using the computer as a tool, *Journal for Research in Mathematics Education* 19 (1988) 3-25.
4. *HP-28S Owner's Manual*, Edition 1, November 1987.
5. *HP-28S Reference Manual*, Edition 1, October 1987.
6. John Selden, Alice Mason, and Annie Selden, Can average calculus students solve nonroutine problems?, *The Journal of Mathematical Behavior* 8 (April 1989) 45-50.
7. Don Small, John Hosack, Kenneth Lane, Computer algebra systems in undergraduate instruction, *The College Mathematics Journal* 17 (November 1986) 423-433.
8. Thomas Tucker, Calculators with a college education?, *Calculus for a New Century: A Pump, Not a Filter*, MAA Notes 8 (1987) 229-231.

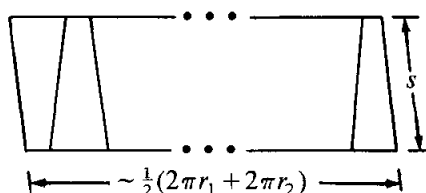
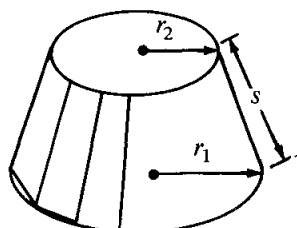
Surface Area of a Cone



$$\text{Area} \sim \pi r^2$$



$$\text{Area} \sim \pi rs$$



$$\text{Area} \sim \pi(r_1 + r_2)s$$

Contributed by Herb Holden, Gonzaga University, Spokane, WA