Applied and
Computational
Mathematics
Division

NISTIR 90-4238

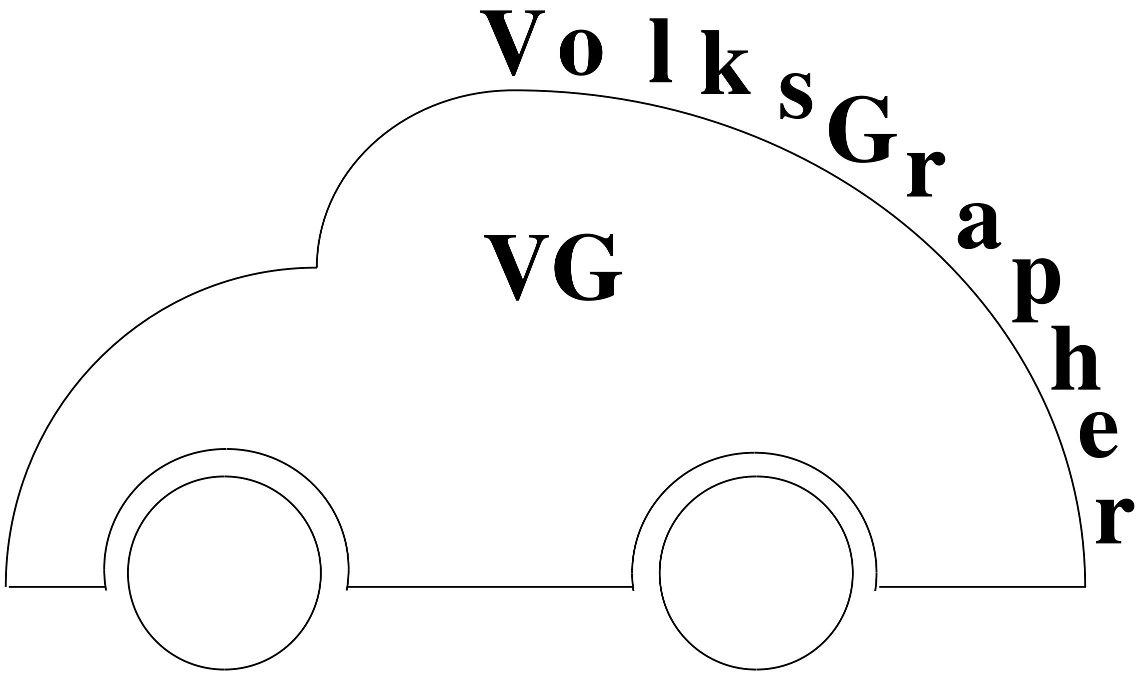
Center for Computing and Applied Mathematics

VolksGrapher: A FORTRAN Plotting Package User's Guide, Version 3.0

D. K. Kahaner and W. E. Anderson

February 1990

U.S. DEPARTMENT OF COMMERCE
National Institute of Standards and Technology
Gaithersburg, MD 20899



VolksGrapher (VG)

VolksGrapher (VG) is a collection of FORTRAN callable subroutines for plotting two dimensional data. The main features of **VG** are:

- Portability—FORTRAN source does not require modification for execution on a variety of computers† including the "PC", Vax, Convex, Irix, and Sun.
- Ease of use—Simple plots require only a few lines of FORTRAN. It is easy to create plots with multiple curves, legends and labels by calling the appropriate **VG** subroutines and linking in the **VG** library. On a PC the screen mode will be automatically recognized.
- Realistic flexibility—**VG** can generate about 90 percent of the routine plots that are of interest to scientists. Nevertheless, the user's manual is short and key ideas can be learned in only a few minutes.

†Certain commercial products are identified. In no case does such identification imply recommendation by NIST, nor does it imply the products are the best available.

- Built-in interactivity—**VG** was designed to be interactive. Without making any changes to the source, users can zoom, insert and drag text and legends around the screen, resize plots, change axes limits, etc. This flexibility is provided automatically by **VG**.

- Support for several printers—**VG** plots can be generated in PostScript, QMS, Tektronix, or HPGL format for publication. On a PC plots can also be "dumped" to an Epson compatible printer.

- Works in the window environment of the Sun—**VG** can produce screen output in a SunView graphics window or on a Tektronix emulation window such as Sun View's tektool or X Window System's xterm.

- **VG** is in the public domain—There are no restrictions on its use. It is designed and maintained by NIST scientists.

VG was the creation of several people at NIST over the last several years. Besides the authors, they include D. Argentar, G. Candela, U. Kattner, and M. Reed. Users of **VG** have not only helped us find bugs ("features") but provided useful suggestions on possible improvements.

The person most responsible for **VG** has been David Kahaner. In his absence from NIST during the next two years, problems with the software should be forwarded to the other author, preferably by electronic mail (anderson@ceee.nist.gov). No significant changes in **VG** are foreseen with the exception of porting it to the X Window System but attempts will be made as time permits to fix any reported and verified bugs.

This document was produced on a Sun by combining the strengths of LaTeX, PostScript, and **VG**. All of the plots were produced by **VG** and saved in PostScript format. The plots were then incorporated into the LaTeXtext file using Trevor Darrell's Psfig macro package (which permits scaling the plots to the desired size) and a TeX macro to rotate them 90°. The PostScript output from the dvi file was then created using Tomas Rokicki's dvips program. This combination of software enables the preparation of camera-ready manuscripts containing graphics without any cutting and pasting.

Table of Contents

List of Figures

1 USAGE

This section briefly describes the necessary commands to link in the **VG** library for various compilers and machines.

For the Lahey F77L version 3.01 compiler on a PC the object file `YOURMAIN.OBJ` can be linked to the **VG** library to produce the executable `YOURMAIN.EXE` with the following command:

```
LINK YOURMAIN,,NUL,VG+F77L
```

making sure `F77L` comes after VG.

For the Ryan McFarland RMFORT compiler version 2.43 on a PC, the link command is:

```
PLINK86 FI YOURMAIN FI FINIT FI VGLOAD LI
        VGRM LI RMFORT
```

(An RMFORT "feature" requires files `FINIT.OBJ` and `VGLOAD.OBJ` to be loaded explicitly.)

For the Suns (SunOs 4.0.x) and SunView support the compile and link command is:

```
f77 -o yourmain yourmain.f -f68881 -Bstatic -lvg
       -lvgsun -lcgi77 -lcgi -lsuntool -lsunwindow
       -lpixrect /usr/lib/f68881/libm.il -lm
```

For the Suns and Tektronix support the compile and link command is:

```
f77 -o yourmain yourmain.f -f68881 -Bstatic -lvg
       -lvgtek /usr/lib/f68881/libm.il -lm
```

For the Suns and batch (noninteractive) support the compile and link command is:

```
f77 -o yourmain yourmain.f -f68881 -Bstatic -lvg
       -lvgbat /usr/lib/f68881/libm.il -lm
```

The above three Sun examples are for the fastest possible execution. The `-Bstatic` flag prevents the use of shared libraries but results in larger executables.

For the Vax running VMS:

```
LINK YOURMAIN,VG/LIB
```

IMPORTANT: Internal routines and COMMON blocks are named `VGxxx`. Don't use such names in your program. Other common errors in using **VG** are:

1. calling a routine with the wrong number of arguments,
2. calling a routine with arguments of the wrong type, e.g. using 4 (integer) when 4.0 (real) is required. Numeric parameter names that begin with I through N are integers; all other variables are real, and
3. forgetting to call `NEWPAG` between pages (or screens) of graphs.

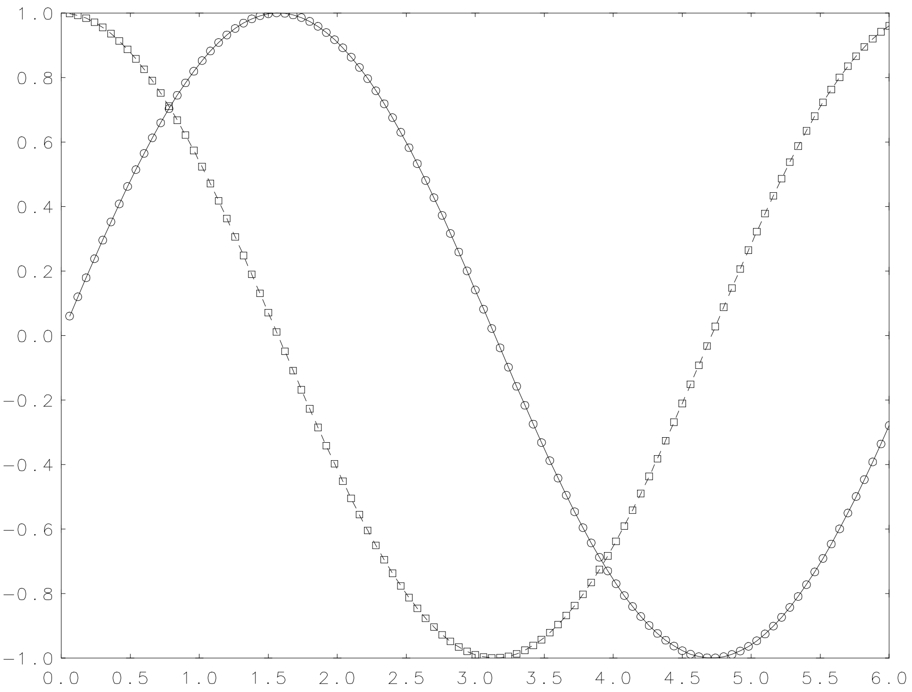


Figure 1. Example **VG** Plot

2 BASIC CALLS

```
      DIMENSION X(100), Y1(100), Y2(100)
      DO 10 I=1,100
         X(I)=0.06*I
         Y1(I)=SIN(X(I))
         Y2(I)=COS(X(I))
10    CONTINUE
20    CONTINUE
         CALL CURV(100, X, Y1)
         CALL CURV(100, X, Y2)
         CALL VG
      IF(LOOPIN() .EQ. 1) GOTO 20
      END
```

This code produces the full screen plot shown in fig. 1 with two curves, each with different lines and symbols, and with axes and axes numbers. After plotting the graph the program will pause leaving you in the interactive mode (see Section 7) awaiting user input. For example, to produce a PostScript file of the screen, press *p*; you will be asked for a file name. (For PC users: To dump the screen to an Epson-type printer, press *d*.) To leave the interactive mode and continue with the next part of your main program, press the *enter* key. A screen (or page) can contain up to six graphs. A graph can contain any number of curves. A screen or page is generated by a program block of the form:

```
XX    CONTINUE
      .
      .
      Call VG
      IF(LOOPIN() .EQ. 1) GOTO XX
```

VG as mentioned above has interactive capabilities. If you use them, `LOOPIN` returns 1, jumping to label `XX`. The code in this loop will be processed a minimum of two times (perhaps several times) depending on the user's interactive input. In general only put code in this loop related to **VG** calls. In any case avoid putting heavy computation inside this loop. For example, code used to generate the plotting arrays should not be placed here. Also avoid putting code in this loop that relies on being executed only once, such as `Y=Y+1` where `Y` is initialized outside the loop. When you press *enter,* `LOOPIN` returns 0, exiting the loop.

Optional calls control graphing symbols, line types, colors, screen position, etc.

The following code will generate a graph on the left and right half of the screen and place one curve in each graph. The resulting plot is shown in fig. 2.

```
      DIMENSION X(100), Y1(100), Y2(100)
      DO 10 I=1,100
         X(I)=0.06*I
         Y1(I)=SIN(X(I))
         Y2(I)=COS(X(I))
10    CONTINUE
C
C   NOTE THAT ONLY CODE RELATING TO VG CALLS IS PLACED IN
C   THE INTERACTIVE LOOP
C
20    CONTINUE
         CALL PLAC(1.0, 0.0, 0.0, 0.5)
C        (TOP, BOTTOM, LEFT, RIGHT)
C        [SCREEN IS UNIT SQUARE, ORIGIN IS BOTTOM LEFT.]
         CALL CURV(100, X, Y1)
         CALL VG
C        ABOVE LINE ENDS FIRST (LEFT) GRAPH
         CALL PLAC(1.0, 0.0, 0.5, 1.0)
         CALL CURV(100, X, Y2)
         CALL VG
C        ABOVE LINE ENDS SECOND (RIGHT) GRAPH
      IF(LOOPIN() .EQ. 1) GOTO 20
      END
```

IMPORTANT: Separate screens or separate plot pages require separate loops, with a call to `NEWPAG` between them.

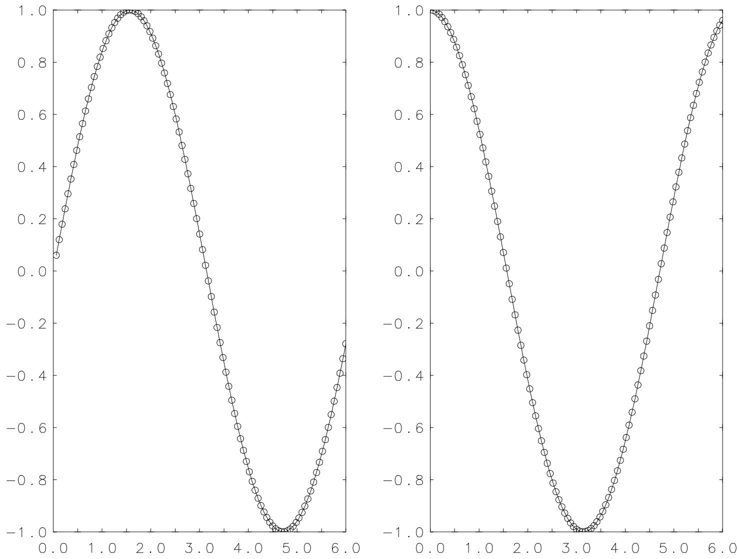


Figure 2. Using `CALL PLAC` for Multiple Plots

The following code will produce two screens. Screen 1 will have a full size plot. Screen 2 will have a plot in the middle third of the page.

```
      DIMENSION X(100), Y1(100), Y2(100)
      DO 10 I=1,100
         X(I)=0.06*I
         Y1(I)=SIN(X(I))
         Y2(I)=COS(X(I))
10    CONTINUE
20    CONTINUE
         CALL CURV(100, X, Y1)
         CALL VG
      IF(LOOPIN() .EQ. 1) GOTO 20
      CALL NEWPAG
30    CONTINUE
         CALL PLAC(0.66, 0.33, 0.33, 0.66)
         CALL CURV(100, X, Y2)
         CALL VG
      IF(LOOPIN() .EQ. 1) GOTO 30
      END
```

3 USING OPTIONAL CALLS

Options can affect the next curve drawn, the current graph, or the current page. Options for the current graph must appear before any calls to `CURV` on that graph. The general form for using options is:

```
       <options affecting the entire page, SETDV,
      1        SETXIT, SCALUP>
XX     <options affecting the first graph, PLAC, AXCODE,
      1        LINLOG, SIDTEX, SETLIM, SETGRD, etc...>
          <options for first curve, HOWPLT, RIGHT, LEGND>
          CALL CURV(<first data set>)

          <optional calls affecting the next curve>
          CALL CURV(<next data set>)
          ...
          CALL CURV(<last data set>)
          CALL VG

          <optional calls affecting the second graph>
          ...
          CALL VG

          (more graphs)
       IF(LOOPIN() .EQ. 1) GOTO XX

       CALL NEWPAG
       (more pages)
       ...
```

The following example produces two graphs on one screen (fig. 3).

```
       DIMENSION X(20), Y1(20), Y2(20), Y3(20)
       DO 5 I=1,20
          X(I)=0.3*I
          Y1(I)=10**X(I)
          Y2(I)=10**((X(I)/2)**2)
          Y3(I)=COS(X(I))**2
5      CONTINUE
C
C      NOTE THAT ONLY CODE RELATING TO VG CALLS IS
C      PLACED IN THE INTERACTIVE LOOP
C
C      HARD COPY TO HP PEN PLOTTER
C
10     CALL SETDV('HPG')
C         OPTIONS FOR FIRST GRAPH
          CALL PLAC(1.0, 0.5, 0.0, 1.0)
          CALL AXCODE(0, 3, 1)
```

```
         CALL LINLOG(0, 1, 1)

C        OPTIONS FOR FIRST CURVE
         CALL HOWPLT(0, 1, 2)
C        FIRST CURVE DATA
         CALL CURV(20, X, Y1)

C        OPTIONS FOR SECOND CURVE
         CALL RIGHT
         CALL HOWPLT(1, 2, 15)
C        SECOND CURVE DATA
         CALL CURV(20, X, Y2)
C        DRAW FIRST GRAPH
         CALL VG

C        SECOND GRAPH OPTIONS
         CALL PLAC(0.5, 0.0, 0.33, 0.66)
C        FIRST CURVE DATA
         CALL CURV(20, X, Y3)
C        DRAW SECOND GRAPH
         CALL VG
      IF(LOOPIN() .EQ. 1) GOTO 10
      ... (program continues)
```

The first graph has two curves, one plotted against the left y-axis, the other against the right y-axis (see `AXCODE` in Section 5.2 and `RIGHT` in Section 4.2). Curve 1 is a solid blue line with no symbols (see `HOWPLT` in Section 4.1); curve 2 is a dashed yellow line with circles at the data points. (On high quality monochrome devices, such as the PostScript output on page 7, curve 2 will be thicker than curve 1). Both y-axes are logarithmic (see `LINLOG` in Section 5.5). The second graph has one curve using the default line type. The axes (x and the left hand y) will be linear; the right hand y-axis is not numbered.

4 OPTIONS AFFECTING A CURVE

4.1 Symbol, Dash Pattern, and Line Color

Unless `HOWPLT` is called, the first curve is plotted with a circle at each data point and solid white lines connecting the points. For subsequent curves, **VG** adds one to the current symbol and dash pattern numbers, and uses the symbol and dash pattern indicated by those numbers. When all the symbols and dash patterns have been used, the color is changed. Each new graph restarts the cycle.

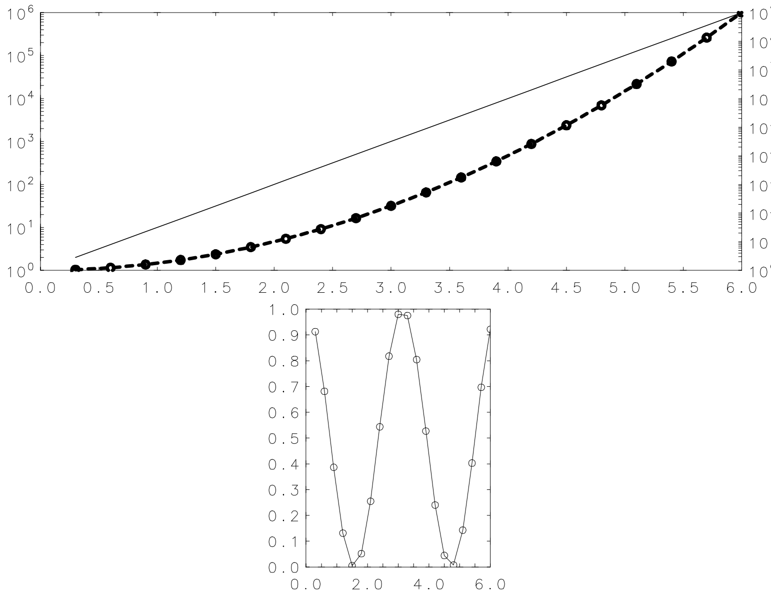


Figure 3. Using `CALL PLAC` with `AXCODE` and `HOWPLT` Options

To set the plotting style of a data set, use:

```
CALL HOWPLT(ISYMBL, ISTYLE, ICOLOR)
```

where `ISYMBL` is the symbol code:

0	no symbol
1–23	various centered symbols shown on the symbol list (fig. 4).

and `ISTYLE` is the dash pattern code:

0	no line
1	solid line
2–11	dashed lines of various patterns (fig. 5)

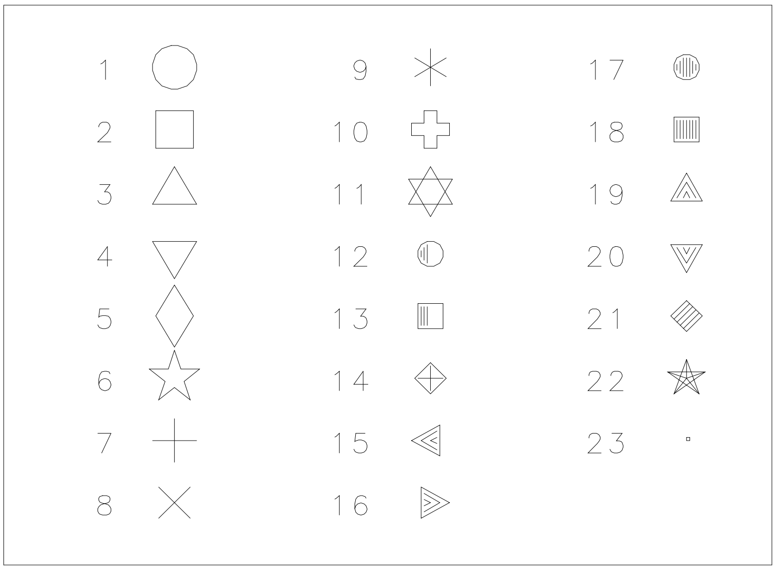


Figure 4. Plot Symbols

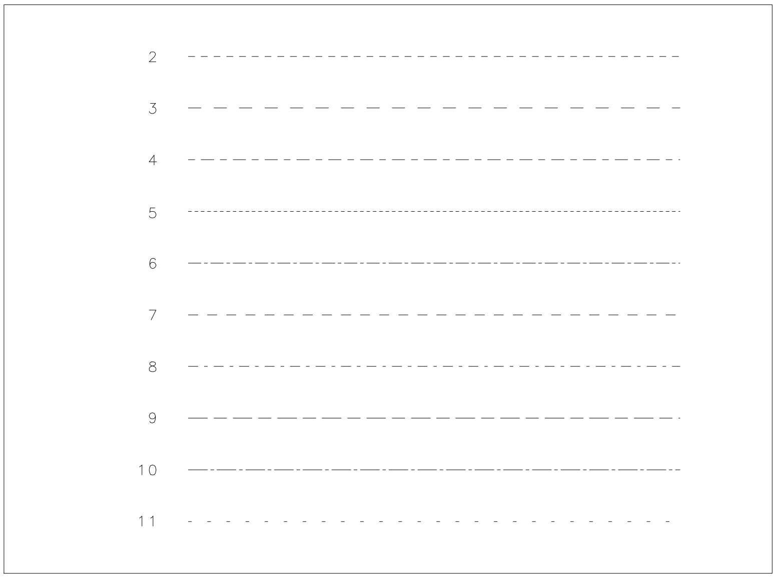


Figure 5. Dash Patterns

and ICOLOR is the color code:

1	intense white	9	dark gray
2	blue	10	light blue
3	green	11	light green
4	cyan	12	light cyan
5	red	13	pink
6	magenta	14	light magenta
7	brown	15	yellow
8	white	16	intense white

Note: your computer may produce fewer colors.

By default a symbol (if ISYMBL is not equal to zero) will be plotted for every data point. To change this behavior use:

```
CALL SETINC(INCSYM)
```

This will result in symbols being plotted every INCSYM data point. (INCSYM is 1 by default).

On laser printers ICOLOR controls the width of the line drawn: larger values of ICOLOR produce wider lines, and on pen plotters ICOLOR controls the pen number.

4.2 Plot Against a Right Hand Axis

To plot data against a right hand axis of a graph, first call AXCODE with IVCODE=3, (see Section 5.2) and then precede the call to CURV with CALL RIGHT. This is illustrated in the example on page 7.

4.3 Legends

A legend is a rectangular box enclosing a list of strings describing each curve on the graph. A sample of the line and symbol used to draw each curve is placed before the string. To generate a legend entry, before calling CURV add the line:

```
CALL LEGND(STRING)
```

Example: CALL LEGND('Water')

To produce multi-line legends call LEGND repeatedly. The legend box is defaulted to the center of its graph. This can be changed (see Section 5.8). The legend can also be moved interactively. There may not be more than fourteen lines in a legend box. An example is shown in fig. 6. Note the use of LEGPOS to position the legend.

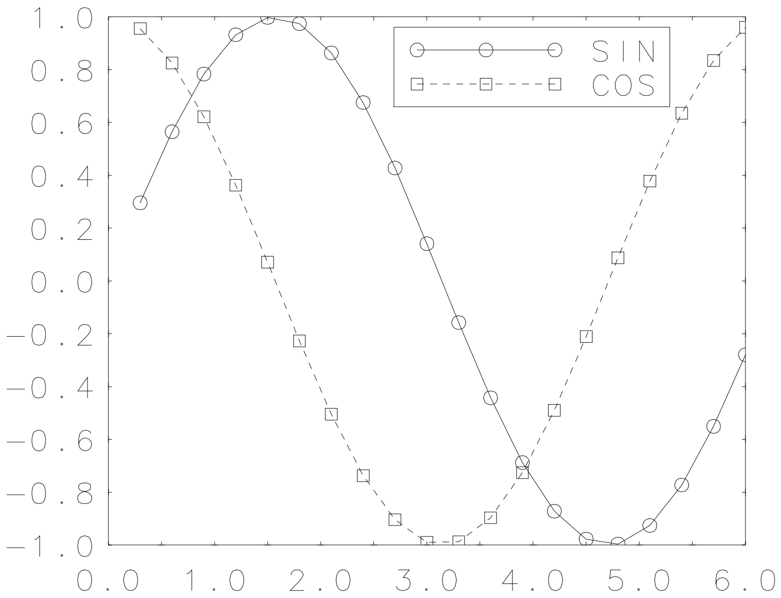


Figure 6. Plots with Legends

```
      DIMENSION X(20), Y(20), Y1(20)
      DO 10 I=1,20
         X(I)=0.3*I
         Y(I)=SIN(X(I))
         Y1(I)=COS(X(I))
10    CONTINUE
      CALL SCALUP(2.0)
20    CONTINUE
         CALL LEGPOS(0.67, 0.88)
         CALL LEGND('SIN')
         CALL CURV(20, X, Y)
         CALL LEGND('COS')
         CALL CURV(20, X, Y1)
         CALL VG
      IF(LOOPIN() .EQ. 1) GOTO 20
      END
```

5 OPTIONS AFFECTING AN ENTIRE GRAPH

5.1 Position on Page

The position of a graph can be changed by using:

```
CALL PLAC(TOP, BOTTOM, LEFT, RIGHT)
```

where 0.0 $\leq$TOP, BOTTOM, LEFT, RIGHT $\leq$ 1.0

The screen is the unit square with origin at lower left.

5.2 Axis Options

The default behavior is to draw the lower and upper horizontal axes, with numbers along the lower axis, and to draw the left and right vertical axes, with numbers along the left axis. To set different options, use:

```
CALL AXCODE(IHCODE, IVCODE, ICOLOR)
```

Where IHCODE is the horizontal axis code number:

0 default horizontal axis option as above
1 lower and upper horizontal axes, BOTH with numbers
2 lower horizontal axis only (with numbers)

and IVCODE is the vertical axis code number:

0 default vertical axis option as above
1 left and right vertical axes, BOTH with numbers
2 left vertical axis only (with numbers)
3 DIFFERENT left and right vertical axes

and ICOLOR is the color of the axes, ticks, and axes numbers (default is 1, see page 9).

Set IVCODE=3 for graphing dependent variables with two different ranges against one independent variable (see Section 4.2).

To suppress numbers on one or more axes, use:

```
CALL NONUM(IX, ILEFTY, IRITY)
```

where IX, ILEFTY, and IRITY are codes for the x, left y, and right y axes: 1 suppresses the numbers on that axis, 0 leaves them on. To suppress the drawing of axes use:

```
CALL NOAXES
```

To suppress drawing of just the x-axis or the y-axis, use CALL NOAXEX or CALL NOAXEY, respectively.

5.3 Axis Limits

The default axis limits are "nice" numbers big enough to contain all the data. To override them, use:

```
CALL SETLIM(IAXIS, ITYPE, AMIN, AMAX)
```

where AMIN and AMAX are the new limits,

IAXIS is the axis to change:

1 x-axis
2 left y-axis
3 right y-axis

and ITYPE is the kind of change:

1 override minimum, but keep "nice" maximum
2 override maximum, but keep "nice" minimum
3 override both minimum and maximum

5.4 Tic Marks

By default tic marks are placed at "nice" numbers and a number is drawn at each tic mark. To change this, use:

```
CALL SETTIC(IAXIS, NTICS, STRARR)
```

Where IAXIS is the axis to change:

1 x-axis
2 left y-axis
3 right y-axis

NTICS is the number of tic marks, and STRARR(1)...STRARR(NTICS) is an array of strings to be placed at the tics. Tics are equally spaced with text centered at each. NTICS must be no greater than 14, and the length of STRARR(I) must be no greater than 20 characters. The tic marks will be unchanged even if the graph is zoomed or the axis limits are changed. If NTICS is negative, |NTICS| tic marks are placed on the the axis, and the precision of each number placed at a tic mark is specified as a string in STRARR. For example, the following fragment:

```
CALL SETLIM(1,3,0.0,3.14159)
CALL SETTIC(1,-3,'4')
```

will produce an x-axis with three tics, labeled "0.0000", "1.5708", "3.1415". In this case, the tic labels will change if the axis limits are changed, or the graph is zoomed.

The following code illustrates the use of SETTIC to produce tic mark labels composed of arbitrary strings (fig. 7).

```
      DIMENSION X(4), Y(4), TICK(6)
      CHARACTER*4 TICK
      DO 5, I=1,4
         X(I)=I
5     CONTINUE
```

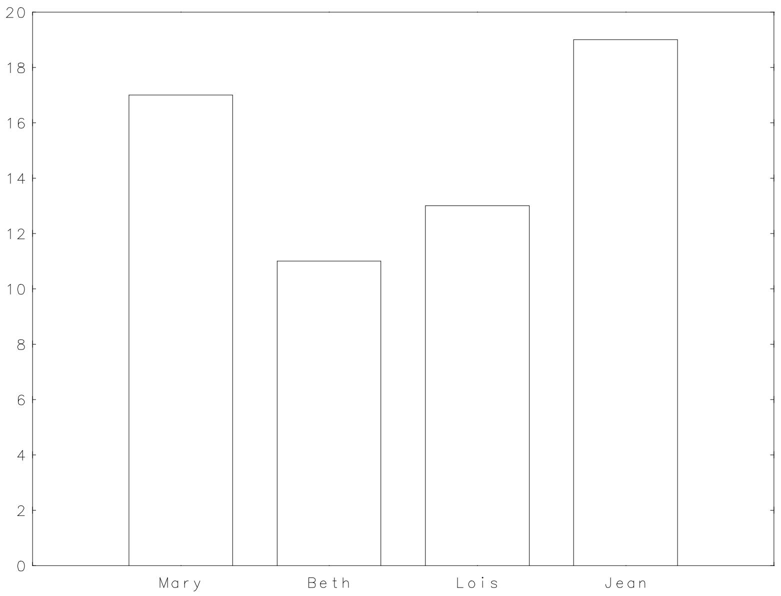


Figure 7. Histogram

```
      Y(1)=17.
      Y(2)=11.
      Y(3)=13.
      Y(4)=19.
      TICK(1)='    '
      TICK(2)='Mary'
      TICK(3)='Beth'
      TICK(4)='Lois'
      TICK(5)='Jean'
      TICK(6)='    '
10    CONTINUE
         CALL HOWPLT(0, 1, 1)
         CALL TICSIZ(0, 0, 10, 10)
         CALL SETLIM(1, 3, 0., 5.)
         CALL SETTIC(1, 6, TICK)
         CALL HIST(4, X, Y, 0.7)
         CALL VG
      IF(LOOPIN() .EQ. 1) GOTO 10
      END
```

`TICSIZ` was used above to suppress the x-axis tic marks (by making them zero length). `TICSIZ` allows the user to change the tic mark length on all four axes. The general form is:

```
      CALL TICSIZ(LTCX1, LTCX2, LTCY1, LTCY2)
```

where LTCX1, LTCX2, LTCY1, and LTCY2 are the tic mark lengths of the lower x, upper x, left y, and right y axes, respectively, and can vary from 0 to 100. The default tic mark length is equivalent to a value of 10. Values outside the acceptable range will result in the default tic mark length.

5.5 Linear or Logarithmic Axes

The axes are linearily scaled by default. To produce a log or semilog graph, use:

```
      CALL LINLOG(ILX, ILY1, ILY2)
```

where ILX, ILY1, and ILY2 are the code numbers for the horizontal axis, the left vertical axis, and the right vertical axis, respectively. In each case, 0 means linear and 1 means log. Unless you call AXCODE with IVCODE=3, (see Section 5.2) ILY2 is ignored. An example is shown on page 7.

5.6 Axis Labels

To draw centered axis labels for a graph, use:

```
      CALL SIDTEX(TPST, ITPC, BTST, IBTC, LFST,
     1                        ILFC, RTST, IRTC)
```

where TPST, BTST, LFST, and RTST are strings to be centered along the top, bottom, left, and right axes, and ITPC, IBTC, ILFC, and IRTC are their respective colors (see color code table on page 9). To omit a string, use ' ' (a blank). Labels created with SIDTEX will be positioned between the graph's axis numbers and borders set by PLAC (see Section 5.1). Note that SIDTEX can produce only single line labels. To produce multi-line labels, use the interactive loop (see Section 7.5), or GTEX (more difficult, but no interaction is needed, see Section 5.7 immediately following). Labels produced with SIDTEX can be changed, moved, etc. interactively like all other text (see Section 7.5). An example of the use of SIDTEX is shown on page 16.

5.7 Placing Text on a Graph

To add text to a graph, use:

```
      CALL GTEX(TEXT, X, Y, ANGLE, SIZE, NCOLOR, MODE)
```

Where

TEXT is the text string, e.g., 'Water',
X is the horizontal coordinate of the first letter of TEXT,
Y is the vertical coordinate of the first letter of TEXT,
ANGLE is the number of degrees to rotate the text string counterclockwise,

`SIZE` is the size of the letters in `TEXT` relative to the base size,
`NCOLOR` is the color code of `TEXT` (see page 9), and
`MODE` describes how the text is placed on the graph. Mode can take on the values:

1: the x and y coordinates are distances in screen units

2: the x any y coordinates are in the same units as the graph's data

3: the text is centered horizontally on the graph, and the y coordinate is a screen distance up from the bottom edge of the graph

4: the text is centered vertically on the graph, and the x coordinate is a screen distance right from the left edge of the graph

5: the text is centered horizontally on the graph, and the y coordinate is a screen distance up from the top edge of the graph

6: the text is centered vertically on the graph, and the x coordinate is a screen distance right from the right edge of the graph

Text characteristics can be changed interactively, but the text will hold its position with respect to its graph (according to its mode) if the graph is moved interactively. That is, the text is "owned" by the graph. (To lock text onto the screen use `HTEX` described in Section 6.1.) To use `GTEX` to make multi-line labels, you'll need to know the height of a line of text (0.02 screen units times the text size, the `SIZE` parameter in `GTEX`, times the scaling factor if you used `SCALUP`, Section 6.2). Each line of text needs to be offset from the graph edge by the width of all text between it and the graph, including numbers on the graph. Also, the left y-axis text is usually rotated to 90.0 degrees and the right y-axis text is usually rotated to -90.0 degrees. The following program will produce a graph (fig. 8) with a three line title, a two line y-axis label, and a one line x-axis label:

```
      DIMENSION X(100), Y(100)
      DO 10 I=1,10
        X(I)=0.6*I
        Y(I)=100.*ABS(SIN(X(I)))
10    CONTINUE
20    CONTINUE
        CALL HOWPLT(0, 1, 1)
        CALL PLAC(1., 0., 0.2, 0.8)
        HEIGHT=0.02
        SIZE=0.02
C     PADDING THREE CHARACTERS WIDE TO AVOID AXIS NUMBERS
        PAD=0.06
        CALL HIST(10, X, Y, 0.2)
C     CREATE TITLE
C       CALL GTEX(TEXT, X, Y, ANGLE, SIZE, NCOLOR, MODE)
        CALL GTEX('Some Data', 0.0, 2*SIZE+0.01, 0.0,
     1            1.0, 1, 5)
```

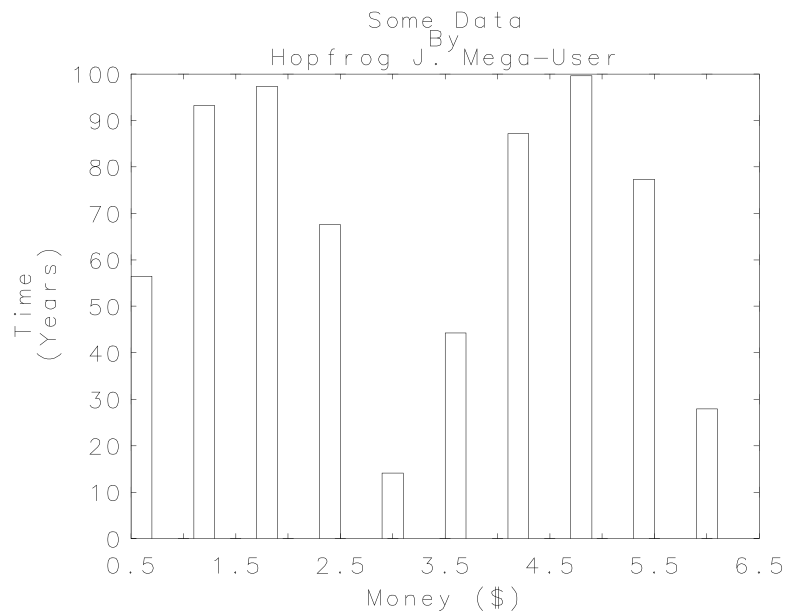


Figure 8. Character Strings in Plots

```
         CALL GTEX('By', 0.0, SIZE+0.01, 0.0, 1.0, 1, 5)
         CALL GTEX('Hopfrog J. Mega-User', 0.0, 0.01, 0.0,
      1               1.0, 1, 5)
C      Y-AXIS LABEL
         CALL GTEX('Time', -HEIGHT-PAD, 0.0, 90.0, 1.0,
      1               1, 4)
         CALL GTEX('(Years)', -PAD, 0.0, 90.0, 1.0, 1, 4)
C      X-AXIS LABEL
         CALL SIDTEX(' ', 1, 'Money ($)', 1, ' ', 1, ' ', 1)
         CALL VG
       IF(LOOPIN() .EQ. 1) GOTO 20
       END
```

There is a limit of 50 lines of text on a page, including text added by `GTEX`, `HTEX`, and `SIDTEX`.

For more information about text, see Section 8.11.

5.8 Legend Placement

The default legend placement is at the center of its graph (see Section 4.3). To change the location for a legend, use:

```
CALL LEGPOS(XLEG, YLEG)
```

where (XLEG, YLEG) are the screen coordinates of the center of the legend. (Screen is unit square.) Thus, CALL LEGPOS(0.5, 0.66) would center a legend halfway across the screen, and two-thirds of the way up. Calling LEGPOS does not prevent a legend from being moved interactively. A legend is surrounded by a box or frame. To make the frame invisible use CALL NOLBOX anytime before the call to **VG**. An example of the use of LEGPOS is shown on page 10.

5.9 Grid Points and Zero Axes

The default is no grid points and no zero axes (lines x=0 and y=0). To change this use:

```
CALL SETGRD(IFID, IZERO)
```

where

IFID=0	no grid points,
IFID=1	draw grid points and
IZERO=0	no zero axes
IZERO=1	draw any zero axis that is on the graph
IZERO=2	draw zero axes only if both are on the graph

5.10 Creating a Blank Box

To place an empty rectangle on a graph, use:

```
CALL BBOX(ATOP, ABOT, ALEFT, ARIGHT, INGRAF)
```

where ATOP, ABOT, ALEFT, and ARIGHT are the sides of the box, and INGRAF=0 indicates that the side coordinates are in screen coordinates, INGRAF=1 indicates that the side coordinates are in the graph's coordinates. The frame can be turned off by using NOLBOX, Section 5.8. A graph cannot have both a legend and an empty box.

6 OPTIONS AFFECTING AN ENTIRE PAGE

6.1 Writing text anywhere on the page

To place text on the page, use:

```
CALL HTEX(TEXT, X, Y, ANGLE, SIZE, NCOLOR)
```

where: TEXT is the text string, e.g. 'water'; X is the horizontal coordinate of the first letter of TEXT (negative value centers, 0.0 is the left edge, 1.0 is the right edge); Y is the vertical coordinate of the first letter of TEXT (negative value centers, 0.0 is the bottom, 1.0 is the top edge); ANGLE is the number of degrees to rotate counterclockwise; SIZE is the size of the letters in TEXT relative to the base size; and NCOLOR is the color code of TEXT (see the color code table on page 9). It is not possible to have more than 50 lines of text on any one screen. This limit includes text added by GTEX, HTEX, and SIDTEX. Text can also be manipulated interactively. Also see Section 8.11.

6.2 Text and Symbol Size

The base height of all text and centered graphing symbols is 0.35 centimeters on a printed page. To change this base size, use:

```
CALL SCALUP(FACTOR)
```

where FACTOR is the desired multiple (or fraction) of the base height. Note that the SIZE parameter in HTEX (see Section 6.1) also affects the height of text, so that the fragment:

```
CALL SCALUP(2.0)
CALL HTEX('A LINE', 0.1, 0.3, 0.0, 1.0, 1)
CALL HTEX('ANOTHER LINE', 0.1, 0.5, 0.0, 0.75, 1)
```

draws “A LINE” at twice normal size and “ANOTHER LINE” at one and a half times normal size.

6.3 Setting the Printer File Format

The default printer file format is PostScript. To produce a file for another type printer, use:

```
CALL SETDV(DEVNAM)
```

where DEVNAM is a three letter code for the printer format:

'tek'	Tektronix
'hpg'	HPGL-Plotter
'pos'	PostScript
'qms'	QMS-Lasergrafix

This remains in effect until changed by another call to SETDV or changed interactively.

7 FEATURES OF THE INTERACTIVE LOOP

When the program pauses after displaying the current page, it is in the normal interactive mode. Typing *enter* causes the program to exit the interactive mode (i.e. when you press *enter* LOOPIN returns 0 resulting in a means to exit the interactive loop). This section describes the interactive commands that can be used.

7.1 Zooming

Type *z* (for "zoom") to enter the zoom mode. Four "zoom corners" will appear in one of the graphs on the screen.

The zoom box corners appear on the "current" graph. Use the space bar to select a different graph. Each time *space* is pressed, a star will flash in the center of the new current graph. Pressing *backspace* selects the previous graph.

Use the cursor control keys to translate the zoom box. (On PCs these are the arrow keys on the right part of your keyboard. On other systems use the letter keys *i, j, k, m* to move up, left, right and down). To move the zoom box in finer increments, type *f;* to go back to coarse movement, type *c* again.

This translates the zoom box; i.e., "whole-moving" mode. Type *s,* to enter "side-moving" mode. Typing a cursor control key causes one side of the box to move outward. To move a side **INWARD**, type a **SHIFTED** cursor control key: e.g., a shifted up arrow or *I* moves the top edge down. To get back to "whole-moving" mode, type *w.*

Once the zoom box has been satisfactorily specified, type *z* again to actually accomplish the zoom.

Type *r,* to "restore" or "zoom out" to the previous graph (i.e. the one prior to the last zoom). You can "zoom out" several times.

Press the *esc* key to remove the zoom box without zooming and return to the normal interactive mode.

Note that zooming will change the mode of all text associated with the graph in mode 2 ("sticking to a point") to mode 1 ("staying a distance from the lower left corner"). See Section 5.7 and Section 8.11.

7.2 Rearranging Graphs

Typing *a* (for "arrange") causes four corners to momentarily flash around the "current" graph and places **VG** in the arrange mode. The space bar, cursor keys, shift keys, *f, c, w,* and *s* work as in zoom mode (see Section 7.1). The current graph will be erased, its new position and shape being indicated only by the corners. To draw the graph in its new position, type *d.* This shifts **VG** back into "normal" mode. To move several graphs, then redraw them all at a stroke: type *a,* use the space bar to select a graph, move it, use the space bar to select a different graph, etc. Then hit *d* and all the graphs you moved will be redrawn.

If the screen becomes cluttered, type *x* which redraws the entire screen.

Press *esc* to leave the arrange mode without changing any graphs.

7.3 Moving Legends

Type *l* (for "legend") to enter legend moving mode. Corners will briefly flash around the "current" legend. Select the legend using the space bar or the backspace key and move the legend around as in the zoom mode. The size of the legend box cannot be changed. Type *d* to draw the currently selected legend, *x* to clean up the screen, or *v* to toggle the current legend between visible and invisible.

Press *esc* to leave the legend mode without changing any legends.

7.4 Editing a Graph's Axes

Type *e* (for "edit") to change the min/max values on a graph's axes, to switch between linear and logarithmic axes, or to number/unnumber axes. Move the pointer $<$ using the cursor keys to select items to change (this does not work too well with Tektronix emulation), and type *c* to change them. Exit edit mode with *esc* which ignores the changes, or with *u* (for "update") to redraw the screen incorporating the changes.

7.5 Altering and Adding Text

Text strings from `HTEX`, `GTEX`, or `SIDTEX`, can be manipulated interactively. New text can be added, even if none of these routines were called. Type *t* to enter the text mode; a star will flash at the first letter of the "current" text string. Use the spacebar or backspace to select a different string. Translate it using the cursor control keys, or rotate it right (clockwise) with *r* or left (counterclockwise) with *l*. The size of the characters can also be changed: they can be made larger by typing $+$ and smaller by typing $-$. Type $>$ to increase the text's color code and $<$ to decrease it (see the table of color codes on page 9). Typing *!* will delete the current text line. To edit the current line, type *e*; you will be prompted for the new contents.

The fineness of translation, rotation and resizing of text can be changed. To go to "fine" movement, type *f;* for "ultrafine", type *u;* to return to "coarse", type *c*.

To change the current string's owner to the current graph, type *o*. The current graph is indicated by corners around its frame. When the screen is the current graph, the corners appear in the screen's corners. To change the current graph, type *]* for the next graph, or *[* for the previous one. (For a discussion of text ownership, see Section 8.11).

To change a current string's mode to "keep a distance" mode, type *d*. To change it to "stick to a point" mode, type *p*. To center the current string horizontally, and change its mode to horizontal centering, type *h*. To center the current string vertically, and change its mode to vertical centering, type *v*. To change a string's mode between normal (left) vertical centering and right vertical centering or normal horizontal (bottom) centering and top horizontal centering, type `@`. (For a discussion of positioning modes, see Section 8.11).

To display the current line's color code, owner number, and positioning mode, type *s* (for "status"). Type *s* again to stop displaying the status.

To make the current centered line into an axis label which will act like one entered with `SIDTEX` (see Section 5.6), type *a*. The current line will then be "owned" by the current graph. The label it becomes depends on its positioning mode: a top centered line becomes a title, and so forth. If the current line is not centered, it will not become a label.

To add new text, type *n* (for "new"). A prompt will appear, asking for the contents of the new line. Entry of the new text line is terminated by pressing the *enter* key. The new line will have the same size, color, and rotation as the current one, and be positioned to line up as the next line "under" it. The new line will have the same positioning mode as the current one, unless the current one is centered. In that case, the new line will have "keep a distance" mode. The new line will be owned by the current graph, **NOT** the current line's owner. The new line becomes the current line, so that blocks of text may be added easily. Note that each page (screen) cannot have more than fifty lines of text. The maximum number of characters in a text string is seventy.

Type *x* to clean up the screen and return to normal mode. Type *esc* to leave text mode without redrawing the screen.

7.6 Printing

VG can produce an output file for a laser printer or pen plotter. **VG** running on a PC can also produce hardcopy output on an attached Epson-compatible printer. To print a graphics screen to an Epson printer, hit *d*; the screen will be "dumped" to the printer. The default format for high quality printing is PostScript (unless altered with `SETDV`, see Section 6.3). This format can be changed in the interactive loop by typing *h* which then prompts for a device name; give the appropriate code name listed on page 18. To print the screen, type *p* and provide a file name. This file can be sent to an appropriate printer. To send the output directly to a printer on a PC, give the name of the printer port (e.g. lpt1) instead of a filename.

7.7 The Cursor

VG has a graphics cursor for finding the position of items on graphs and extracting those values into your program. Type *s* to show the cursor. Then type *w* to display the cursor position in coordinates of the current graph. Typing * passes the cursor's location by calling `VGCURS(X, Y)`. To capture this data, write `SUBROUTINE VGCURS(X, Y)` and save local values of X and Y. For example, the following program fragment will log up to six cursor positions and then take their average:

```
      REAL SAVEX(6),SAVEY(6),SUMX,SUMY,AVGX,AVGY
      COMMON/CDATA/SAVEX,SAVEY,I
      ...
      I=0
10    CALL PLAC(1.0, 0.5, 0.0. 0.5)
         ...(OTHER VG CALLS)...
      CALL VG
      IF(LOOPIN().EQ.1)THEN 10
C     TAKE AVERAGE OF VALUES LOGGED BY CURSOR
      SUMX=0.0
      SUMY=0.0
      IF (I.GT.0) THEN
         DO 20 J=1,I
            SUMX=SUMX+SAVEX(J)
20       SUMY=SUMY+SAVEY(J)
         AVGX=SUMX/I
         AVGY=SUMY/I
      ENDIF
      ...

      SUBROUTINE VGCURS(X, Y)
      REAL SAVEX(6),SAVEY(6)
      COMMON/CDATA/SAVEX,SAVEY,I
      I=I+1
      IF(I.LE.6)THEN
         SAVEX(I)=X
         SAVEY(I)=Y
      ENDIF
      RETURN
      END
```

If SUBROUTINE VGCURS(X, Y) does not exist, typing * will do nothing. To toggle cursor off, type *s* again.

7.8 Leaving the Interactive Loop

Leave the interactive loop by pressing the *enter* key. The calling program continues after resetting the screen to text mode. A second keystroke to end the loop can be added with SETXIT (see Section 8.2).

A sequence of plots can be converted to high quality print files without interaction, by typing *g* (for "go"). The user will be prompted for a plot file name such as `PLOTS`. The function `LOOPIN()` will return 0 for the remainder of the program preventing access to the **VG** interactive mode. If the high quality format is Postscript, the files will be called `PLOTS01.POS`, `PLOTS02.POS`, etc. Also see Section 8.4.

If you have set automatic printing (see Section 8.3), and want to leave without creating a print file for the current screen, type *o*, (for "override"). Overriding for one screen will not turn off automatic printing for subsequent screens.

8 MISCELLANEOUS FEATURES

8.1 Adding a Comment Line to the Loop

To provide a comment line which will not appear when the screen is printed (to provide instructions to the user, for instance) use:

```
CALL COMMEN(STRING)
```

For example: `CALL COMMEN('<Enter> to Exit').`

The comment line will be hidden when you enter any of the modes in the interactive loop. The comment line will be in effect until changed. To remove the comment line, replace `STRING` with a blank: `CALL COMMEN(' ').` The comment line is left justified.

8.2 Adding a Second Exit Character to the Loop

In addition to the normal loop exit character, *enter*, a second character can be established. The second exit character is specified by:

```
CALL SETXIT(ICHAR)
```

For example: `CALL SETXIT(27)`

where `ICHAR` is the ASCII code for the character. The code of a common exit character, *esc*, is 27. The `SETXIT` remains in effect until changed. Calling `SETXIT(0)` forces **VG** to exit immediately after flashing a plot, without any user input.

8.3 Printing on Loop Exit

To have **VG** automatically create a high quality print file whenever you exit the interactive loop, place

```
CALL XITPRN(.TRUE.)
```

before the call to LOOPIN. When you exit any screen after CALL XITPRN(.TRUE.), **VG** will create the print file VGnn.dev (where nn is the page number of that page, and dev is the three letter code for the printing format). To change "VG" to another name see SETFIL in Section 8.4. Automatic printing will remain in effect until turned off with:

```
CALL XITPRN(.FALSE.)
```

Automatic printing can be overridden from the interactive loop to allow exit without printing (override with *o,* see Section 7.8).

8.4 The Batch Loop

The batch loop is used for creating high quality output files without any interaction and without the requirement of having a supported screen device. Batch operation requires replacing LOOPIN with LOOPBT.

The default output file is VGnn.dev, where nn is the page number of that output page, and dev is the three letter code for the printer format. To change VG to some other name,

```
CALL SETFIL(NAME)
```

where NAME is the base file name of up to six characters. **VG** appends the page number and the device type to the name, so it is not possible in this case for PC users to specify output to go directly to an attached printer. No more than 99 pages can be generated in batch loops. A main program can contain calls using both the interactive and batch loops.

8.5 Smooth Interpolation

Normally, **VG** plots an optional symbol at each data point (unless changed by SETINC, see page 9), connecting these symbols with optional straight line segments. To produce a smooth curve through a data set replace the call to CURV with a call to SCURV. The interpolation scheme† was developed by Dyn, Gregory, and Levin and has been referred to as a 4-point subdivision scheme similar to approximations using Beziér functions. For N (the number of data points) less than 3 or N greater than 847 there will be no interpolation. An example is shown in fig. 9.

8.6 Contour Plots

In order to create a contour plot, replace a call to CURV with:

```
CALL CONTOR(N, M, X, Y, Z)
```

†*Numerical Methods and Software* by David Kahaner, Cleve Moler, and Stephen Nash. See particularly problem 4-11 on page 131.

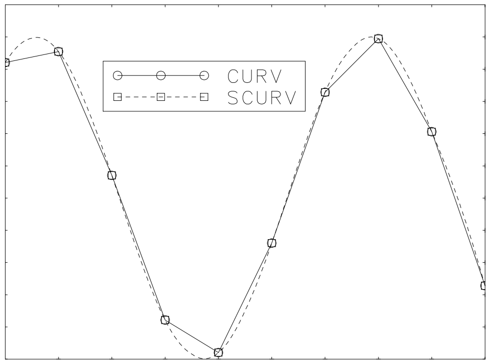


Figure 9. Smooth Interpolation

where:

- `N` is the number of columns of points,
- `M` is the number of rows of points,
- `X` is a vector containing the x-values of the columns,
- `Y` is a vector containing the y-values of the rows, and
- `Z` is an `N` by `M` array containing the z-values at each point.

There may be only one contour plot per graph, and a contour plot may not have any other curves (from `CURV`, `SCURV`, `ERRBAR`, or `DCURV`) on it. Having a contour plot does not affect the options available to other graphs. To change options on the z-axis (`LINLOG` or `SETLIM`, for example) refer to it as the right hand y-axis. For instance,

```
      CALL SETLIM(3, 1, 5.0, 0.0)
      CALL CONTOR(N, M, X, Y, Z)
      CALL VG
```

sets the contour plot's z-axis minimum to 5.0. `CONTOR` adds a legend to the contour plot describing the min, max, and contours of the plot. More legend lines can be added to this legend. To suppress the `CONTOR` legend lines, turn off numbering on the z-axis (see Section 5.2 and Section 7.4).

The following example illustrates the use of `CONTOR` (fig. 10).

```
      DIMENSION X(41), Y(41), Z(41,41)
      DO 10 I=1,41
         X(I)=0.03*(I-21)
         Y(I)=0.03*(I-21)
10    CONTINUE
      DO 30 I=1,41
         DO 20 J=1,41
            R=SQRT(X(J)**2+Y(I)**2)
            Z(I,J)=EXP(-R**2)
20       CONTINUE
```

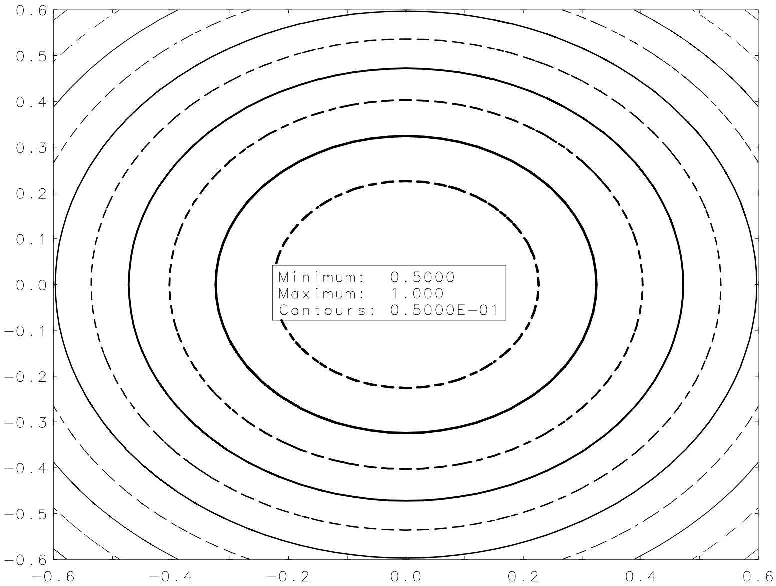


Figure 10. Contour Plots

```
30    CONTINUE
40    CONTINUE
         CALL CONTOR(41, 41, X, Y, Z)
         CALL VG
      IF(LOOPIN() .EQ. 1)GOTO 40
      END
```

8.7 Error Bars

To plot a curve with error bars replace a call to `CURV` with:

```
      CALL ERRBAR(N, X, Y, XERRS, YERRS, IX, IY)
```

where:

`N`, `X`, `Y` have the same meaning as in `CURV`,
`XERRS` is a vector of errors in the x-direction,
`YERRS` is a vector of errors in the y-direction, and
`IX=1` indicates error bars in `X` are to be drawn,
`IX=0` indicates no error bars in `X` are to be drawn,

Similarly `IY` can equal 0 or 1. An example is shown in fig. 11.

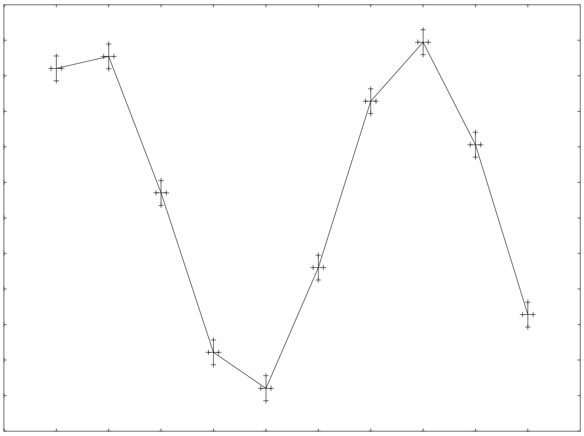


Figure 11. Error Bars

8.8 Histograms

To plot a histogram, replace a call to `CURV` with:

```
CALL HIST(N, X, Y, BARWID)
```

where `N`, `X`, `Y` have the same meaning as in `CURV`, and `BARWID` is the width of the bars drawn in the user's units. Examples are shown on pages 13 and 16.

8.9 Double Precision Data

To plot a curve with double precision data, replace `CURV` with `DCURV`. The data is internally converted to REAL, so that `DCURV` is bounded by the limits of REALs (approximately 10^{-37} to 10^{37} on a PC.

8.10 How VG Handles Errors

VG, if possible, ignores incorrect input. For example if `HOWPLT` is called as

```
CALL HOWPLT(1, 23, 4)
```

the default line type (color, dash pattern, and symbol) is used. **VG** will use the defaults if reals are replaced by integers.

8.11 Text in VG

There are several ways to create lines of text in **VG**: `SIDTEX` (Section 5.6) creates one-line labels for graphs, `GTEX` (Section 5.7) attaches text to graphs, `HTEX` (Section 6.1) makes text elsewhere, and text can be added in the interactive loop. To support functions such as the centering of axis labels and the positioning of text, information about each line of text is needed beyond its location, size, and color. These pieces of information are the line's owning graph and its positioning mode.

All text is "owned" by a graph. When the line's owner is moved or changed, the text responds: if a graph is moved, its title becomes centered over its new location. Owner numbers are assigned to graphs in the order the graphs are created in your program. To avoid having to make a special case, the screen is considered to be a graph, and is given the owner number 0. When text is made with `HTEX`, its owner is the screen. When text is made with `GTEX` or `SIDTEX`, its owner is the graph which made it.

A text line's positioning mode describes how the line reacts to changes in its owner. There are six positioning modes. (1) Keep the line a given distance from the lower left corner of the owner, regardless of how the owner's size or axis limits change. (2) Keep the line at a given coordinate on the graph, even when the graph is resized or given new axes limits. When the owner is the screen, there is no difference between (1) and (2). The last four modes deal with centering: (3) centers horizontally with respect to the owner's edges, staying a constant distance up from the owner's bottom edge; (4) centering vertically, offset from the left edge; (5) centers horizontally, like (3), but the offset is from the top edge; and (6) centers vertically, similar to (4), but the offset is from the right edge. `SIDTEX` sets positioning modes to keep the labels correctly centered; `HTEX` sets the positioning mode implied by by the input. `GTEX` takes the desired mode as a parameter.

Both owner and positioning mode can be changed interactively. Only centering results in a visible change in the text, but the effects often will not be seen until graphs are changed or moved. A status line is provided, giving the text's owner number and mode, so one can tell how the text will act.

8.12 Determining Version Number of VG Library

To determine the version of **VG**, use

```
CALL VGVER
```

A text string of the form `VG VERSION 3.08 -- 24 February 90` will be displayed.

8.13 Greek, Superscript, and Subscript Characters

To draw a Greek character, precede the letter by `|`. Thus `|a` will draw an alpha. To subscript or superscript a character, precede it with _ or `^`. Thus `H_2O` is the formula

Figure 12. Greek Characters

for water `H`$_2$`O`. Use `^^` to get super-superscript, like `e^x^^2` (e^{x^2}). To generate a Greek subscript, such as alpha, use `_|a`. To generate `|`, `_`, or `^` precede it with `|` as in `||sin(x)||`. The available Greek characters are shown in fig. 12.

8.14 Special Characters

Some extra characters of occasional use (e.g. integral, square root, or arrow) are also available. To access them first lookup their **THREE** digit code in figs. 13 or 14 (for example the code for a right arrow is 169). Then in any call to `HTEX`, `GTEX`, or `SIDTEX`, or interactively, enter the code as a string preceded by a "\", for example \202. To get a backslash, use \220.

000	○	032		064	@	096	`
001	□	033	!	065	A	097	a
002	△	034	"	066	B	098	b
003	▽	035	#	067	C	099	c
004	◊	036	$	068	D	100	d
005	☆	037	%	069	E	101	e
006	+	038	&	070	F	102	f
007	×	039	’	071	G	103	g
008	∗	040	(	072	H	104	h
009	✙	041	)	073	I	105	i
010	✡	042	*	074	J	106	j
011	◐	043	+	075	K	107	k
012	◧	044	,	076	L	108	l
013	◈	045	−	077	M	109	m
014	◀	046	.	078	N	110	n
015	▶	047	/	079	O	111	o
016	●	048	0	080	P	112	p
017	■	049	1	081	Q	113	q
018	▲	050	2	082	R	114	r
019	▼	051	3	083	S	115	s
020	◆	052	4	084	T	116	t
021	★	053	5	085	U	117	u
022	·	054	6	086	V	118	v
023		055	7	087	W	119	w
024		056	8	088	X	120	x
025		057	9	089	Y	121	y
026		058	:	090	Z	122	z
027		059	;	091	[	123	{
028		060	<	092	\	124	|
029		061	=	093	]	125	}
030		062	>	094	^	126	~
031	☺	063	?	095	_	127	

Figure 13. Regular Characters

128	○	160		192	§	224	`
129	□	161		193	A	225	α
130	△	162		194	B	226	β
131	▽	163	≠	195	H	227	η
132	◊	164	£	196	Δ	228	δ
133	☆	165	÷	197	E	229	ε
134	+	166		198	Φ	230	φ
135	×	167	°	199	Γ	231	γ
136	∗	168	←	200	X	232	χ
137	✙	169	→	201	I	233	ι
138	✡	170	∞	202	∫	234	∂
139	◐	171	±	203	K	235	κ
140	◧	172		204	Λ	236	λ
141	◈	173		205	M	237	μ
142	◀	174	¨	206	N	238	ν
143	▶	175	√	207	O	239	ο
144	●	176		208	Π	240	π
145	■	177		209	Θ	241	ϑ
146	▲	178		210	P	242	ρ
147	▼	179		211	Σ	243	σ
148	◆	180		212	T	244	τ
149	★	181		213	ϒ	245	υ
150	·	182		214	Å	246	ß
151		183		215	Ω	247	ω
152		184		216	Ξ	248	ξ
153		185		217	Ψ	249	ψ
154		186	⇌	218	Z	250	ζ
155		187	≃	219		251	
156		188	≤	220	\	252	|
157		189	≡	221		253	
158		190	≥	222		254	≈
159	☺	191	?	223	‾	255	

Figure 14. Special Characters

9 SUMMARY

9.1 Summary of VG Calls

Call	Summary	Page
AXCODE	set axis options	11
BBOX	create a blank box	17
CURV	draw curve on a graph	2
COMMEN	place a non-printing comment line	23
CONTOR	draw a contour plot	24
DCURV	draw curve using double precision	27
ERRBAR	draw curve with error bars	26
GTEX	add text to a graph	14
HIST	draw a histogram	27
HOWPLT	set line and symbol types for curve	6
HTEX	draw line of text of any character size	17
LEGND	produce legend entry for curve	9
LEGPOS	set position for legend	16
LINLOG	set linear or logarithmic axes for graph	14
LOOPBT	call batch loop	24
LOOPIN	call interactive loop	2
NEWPAG	start new screen (page)	2
NOAXE{S,X,Y}	suppress drawing of graph's axes	11
NOLBOX	suppress drawing box around legend	17
NONUM	suppress drawing of graph's axis numbers	11
PLAC	set graph's location	10
RIGHT	plot next curve against right y-axis	9
SCALUP	change size of characters and symbols	18
SCURV	draw smoothed curve	24
SETDV	set high quality syntax	18
SETFIL	set batch loop high quality file name	24
SETGRD	set grid options	17
SETINC	change "distance" between plot symbols	9
SETLIM	set axis limits	11
SETTIC	set number of tic marks and tic text	12
SETXIT	add another interactive loop exit key	23
SIDTEX	draw axis labels	14
TICSIZ	modify length of tic marks	13
VG	draw all features of graph, excluding curves	2
VGCURS	get cursor data	21
VGVER	print **VG** version number	28
XITPRN	set printing on interactive loop exit	23

9.2 VG Call Parameters

Call with Parameters

9.3 Interactive Loop Commands

Command	Effect
Normal mode commands	
space	move to next graph
backspace	move to previous graph
enter	leave interactive loop, setting screen to text mode
a	enter graph arrange mode (see below for options)
c	switch to coarse cursor movement
e	enter graph editing mode (see below for options)
f	switch to fine cursor movement
g	leave interactive loop, screen to text mode, flash next plots
h	set high quality output device (pos, qms, hpg, tek)
l	enter legend moving mode (see below for options)
o	leave interactive loop, overriding automatic printing
p	create high quality graphics output file
r	restore current graph by unzooming one level
s	show cursor
t	enter text mode (see below for options)
w	display cursor coordinates
x	redraw (cleanup) screen
z	enter zoom mode (see below for options)
*	catch cursor's position
Arrange mode commands	
esc	exit arrange mode, making no changes
space	move to next graph
backspace	move to previous graph
cursor keys	adjust position of graph [also use *ijkm* keys]
shift	modifies effect of cursor keys in side mode
c	switch to coarse movement
d	exit arrange mode, drawing the current positions
f	switch to fine movement
s	enter graph side moving mode
w	enter whole graph moving mode
Edit mode commands	
esc	exit edit mode, making no changes
cursor keys	move pointer [also use *ijkm* keys]
c	change current item
u	update graph to reflect change and exit

Command	Effect
Legend mode commands	
space	move to next legend box
backspace	move to previous legend box
cursor keys	adjust position of legend box [also use *ijkm* keys]
c	switch to coarse movement
d	draw the currently selected legend, leave legend mode
f	switch to fine movement
v	toggle legend between visible and invisible
Text mode commands	
space	move to next text line
backspace	move to previous text line
cursor keys	move current text line [also use *ijkm* keys]
a	make current line into an axis label
c	switch to coarse movement
d	change current line's mode to "keep distance"
e	edit current line of text (*e esc enter* deletes line)
f	switch to fine movement
h	center current text line horizontally
l	rotate current text line left (counterclockwise)
n	add a new text line
o	change current text line's owner to current graph
p	change current line's mode to "stick to point"
r	rotate current text line right (clockwise)
s	show color, owner, and mode of current text line
u	switch to ultrafine movement
v	center current text line vertically
x	exit text mode, redrawing screen
+	increase size of current text line
-	decrease size of current text line
!	delete current text line
$<$	decrease current text line's color code
$>$	increase current text line's color code
[	move to previous graph
]	move to next graph
@	change current line's centering mode

Command	Effect
Zoom mode commands	
esc	exit zoom mode, making no changes
cursor keys	adjust position of zoom box [also use *ijkm* keys]
shift	modifies effect of cursor keys in side mode
c	switch to coarse movement
f	switch to fine movement
s	enter zoom box side moving mode
w	enter whole zoom box moving mode
z	zoom, leaving zoom mode

10 Examples

Several examples are shown on the following pages of graphs produced by **VG**. The authors wish to thank Ray Mountain, Cheol Park, Jerry Stenbakken, and Charles Fenimore for providing them.